UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Um Método para Auxílio do Ensino de Simulação de Redes de Computadores Apoiado em UML

YASUE KASHIMURA

MANAUS, SETEMBRO DE 2006

UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

YASUE KASHIMURA

Um Método para Auxílio do Ensino de Simulação de Redes de Computadores Apoiado em UML

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO PROGRAMA DE PÓS-GRADUAÇÃO (STRICTO SENSU) EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO AMAZONAS COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO.

Aprovado por :

Orientador : ______________________________

Edjair de Souza Mota , Dr.-Ing.

Banca examinadora :

____

Edson do Nascimento Silva Jr., Dr.

____

José Neuman de Souza , Dr.

____

Edjard de Souza Mota , PhD

MANAUS, SETEMBRO DE 2006

Aos meus pais

Motoyasu Kashimura e Shizuko Kashimura

Agradecimentos

À minha mãe pelo incentivo e apoio ao longo da minha vida.

Ao meu orientador Prof. Edjair de Souza Mota, pela orientação para a realização deste trabalho.

À Fundação Centro de Análise, Pesquisa e Inovação Tecnológica (FUCAPI), pela oportunidade de cursar este Programa de Pós-Graduação em Informática.

Ao colega Laércio Júnior pela colaboração na especificação do estudo de caso deste trabalho.

Aos professores Edson do Nascimento, José Neuman e Edjard Mota pela presença na banca examinadora.

Resumo

A comunicação entre dispositivos portáteis está tornando-se essencial no nosso cotidiano, proporcionando mobilidade e acesso a serviços independentemente de onde estão localizados. A grande diversidade de elementos tais como agentes de conexão, protocolos de roteamento, alterações topológicas e medição da qualidade de serviço da rede envolvidos nesse ambiente tornam a modelagem dos problemas em redes *Ad Hoc* uma tarefa difícil. A simulação é uma atividade importante, nesse contexto, porém existem dificuldades quanto ao processo de elaborar modelos de simulação. A área da Engenharia de Software pode miminizar as dificuldades encontradas na utilização de ferramentas de simulação. O objetivo deste trabalho é apresentar um método para auxiliar o ensino de simulação de redes de computadores a partir de modelos de cenários representados através das técnicas de modelagem empregadas na Engenharia de Software. Estas técnicas envolvem diagramas e mecanismos de extensões encontrados em ferramentas de modelagem, que permitem a conversão dos modelos descritos em UML no padrão XMI, proporcionando a geração de um *script* OTcl na sintaxe do Network Simulator. Além disso, espera-se que este trabalho auxilie na metodologia para descrever um cenário para simulação de redes *Ad Hoc* especificamente.

Abstract

The communication between portable devices is becoming essential in our life, offering mobility and access to services independent of where are located. The large diversity of members such as agents, routing protocols, topologies changes and measure of services of quality of network involved in this environment become the modeling of the problems in *Ad Hoc* networking a hard task. The simulation becomes an important task in this context, however there are difficulties as for the process to elaborate models of simulations. Software Engineering can minimize the difficulties found in the use of simulation tools. The purpose of this work is to propose an environment to assist the teaching of simulations of systems from models of networking being represented through of the modeling techniques employed in the Software Engineering. These techniques involve diagrams and extension mechanisms used in modeling tools that let a conversion the models described in UML to XMI pattern, giving a generation of a OTcl *script* in the syntax of the network simulator. Besides, it expects this work assists in the methodology to describe a scene to simulation of *Ad Hoc* networks.

Sumário

Lista de Figuras

LISTA DE FIGURAS

Lista de Tabelas

LISTA DE TABELAS

Lista de Abreviaturas e Siglas

AODV : *Ad Hoc Distance Vector;*
ACK : *Acknowlegdement;*
CBR : *Constant Bit Rate;*
DSDV : *Destination Sequenced Distance Vector;*
DSR : *Dynamic Source Routing;*
FTP : *File Transfer Protocol;*
IEEE : *Institute for Electrical and Electronic Engineers;*
LAN : *Local Area Network;*
MAC : *Medium Access Control;*
MONARCH : *Mobile Networking Architecture;*
MOS: *Mean Opinion Score;*
NAM : *Network Animator;*
ns: *Networking Simulator;*
Tcl : *Tool command Language;*
TCP: *Transport Control Protoco;*
OCL: *Object Constraint Language;*
OMG : *Object Management Group;*
PDA : *Personal Digital Assistants;*
PERL: *Pratical Extraction and Report Language;*
RED: *Random Early Discard;*
RTP: *Real-Time Transport Protocol;*
SFQ: *Stochastic Fair Queueing;*
SVG: *Scalable Vectorial Graphics;*
UDP : *User Datagram Protocol;*
UML : *Unified Modeling Language;*
TORA : *Temporally-Ordered Routing Algorithm;*

XMI : *eXtended Meta Language Interchange;*

Capítulo 1
Introdução

A comunicação entre dispositivos portáteis está, cada vez mais, tornando-se essencial no nosso cotidiano, proporcionando mobilidade e acesso a serviços independentemente de onde estão localizados. Uma arquitetura de rede que se propõe a dar suporte a essa mobilidade, sem a necessidade de elementos centralizadores (como pontos de acesso), é chamada de rede *Ad Hoc* ou espontânea. A comunicação é feita diretamente entre os terminais móveis, que também atuam como roteadores quando o terminal móvel de destino está fora do alcance do terminal móvel de origem.

A grande diversidade de elementos tais como agentes de conexão, protocolos de roteamento, alterações topológicas e medição da qualidade de serviço da rede envolvidos nesse ambiente tornam a modelagem dos problemas em redes *Ad Hoc* uma tarefa difícil, exigindo a utilização de técnicas para avaliar o desempenho de uma rede *Ad Hoc* considerando esses elementos.

Uma das técnicas mais usadas é a simulação que consiste em realizar experimentos em um modelo do mundo real. Experimentos usando um sistema real são, sem dúvida, complicados e dispendiosos, implicando em grandes gastos e riscos inesperados. Dessa forma, a simulação permite um grau de abstração que pode ser do mais simples ao mais detalhado [9] dependendo do modelo a ser construído, conduzindo à sua melhor compreensão e às sugestões para melhorá-lo. As variações de cenários, isolamento de parâmetros e a exploração de métricas usadas em simulações fornecem aos pesquisadores resultados com a precisão desejada. Além disso, a introdução de novos componentes no modelo simulado pode auxiliar na previsão de áreas problemáticas durante a operação do sistema.

A simulação permite estudar e experimentar complexas interações internas de um dado sistema, identificando os efeitos que certas variações podem causar na organização e das informações relativas à operação desse sistema, antes de serem executados no ambiente real. A simulação pode ser utilizada como material pedagógico, para ensinar habilidades básicas na análise teórica, na análise estatística e nas decisões, tanto a estudantes como a profissionais [9]. O uso de cenários e simulações possibilita ao usuário reunir estatísticas significativas sobre a operação do sistema que será desenvolvido. Às vezes, esse processo de desenvolvimento pode resultar em um produto quase pronto e preparado para ser executado no ambiente real.

Atualmente, ferramentas de simulação são desenvolvidas para automatizar o processo de simular experimentos. Em redes *Ad Hoc*, as ferramentas de simulação são bastante utilizadas e devem lidar com inúmeros requisitos e restrições aplicados para resolver problemas neste tipo de rede, dificultando o seu entendimento e até mesmo a sua utilização por estudantes e pesquisadores. A área da Engenharia de Software pode miminizar as dificuldades encontradas na utilização de ferramentas de simulação, pois esta área tem aprimorado técnicas e metodologias que proporcionam a concepção e a construção de sistemas com mais qualidade e produtividade. Uma dessas técnicas é a UML (Unified Modeling Language) [16], uma linguagem visual para modelagem de sistemas que pode auxiliar no processo de simulação de experimentos através da modelagem de cenários de redes *Ad Hoc*.

1.1 Motivação

As ferramentas de simulação de redes de computadores são importantes para proceder análises e avaliações de experimentos em projetos no Labortatório de Redes da UFAM (Universidade Federal do Amazonas). Uma das ferramentas bastante utilizada no meio acadêmico é o Network Simulator, conhecido como *ns-2*. O usuário escreve um *script* com os elementos do seu modelo do cenário de rede *Ad Hoc* para simular o experimento. O *script* é um programa que faz uso da linguagem OTcl (uma extensão orientada a objeto para

linguagem Tcl) e os elementos devem ser escritos na sintaxe desta linguagem. Uma das carências apresentadas nesse procedimento para criar uma simulação no *ns-2* é a dificuldade de manipular as sintaxes que correspondem ao modelo de simulação de problemas de rede para gerar experimentos.

A ferramenta Network Simulator possibilita simulações para vários tipos de redes de computadores, porém o usuário pode consumir mais tempo pesquisando e coletando informações sobre a sintaxe do simulador para escrever um modelo de simulação em *ns-2* (um *script* OTcl) específico do seu experimento. Em muitos casos, ele necessita mudar o foco do seu esforço para entender melhor o simulador, afastando-se da questão sendo investigada através do experimento.

Observou-se que os experimentos são feitos por usuários que têm conhecimentos em UML. Através de uma linguagem visual como o UML, o usuário pode capturar os elementos mais relevantes do seu cenário para produzir um modelo de experimento para simulação.

1.2 Objetivo

O objetivo deste trabalho é apresentar um método para auxiliar o ensino de simulação de redes de computadores a partir de modelos de cenários representados através das técnicas de modelagem empregadas na Engenharia de Software. Estas técnicas envolvem diagramas e mecanismos de extensão encontrados em ferramentas de modelagem, que permitam a conversão dos modelos descritos em UML no padrão XMI (eXtended Meta Language Interchange) [13], proporcionando a geração de um *script* OTcl na sintaxe do Network Simulator. Além disso, espera-se que este trabalho auxilie na metodologia para descrever um cenário para simulação de redes *Ad Hoc* especificamente.

Este trabalho está dividido em cinco Capítulos. O Capítulo 2 apresenta uma revisão dos temas abordados neste trabalho que envolve redes *Ad Hoc*, simulação de redes de

computadores e Engenharia de Software. O Capítulo 3 apresenta uma proposta para modelagem de simulação de redes *Ad Hoc* através dos diagramas UML. O Capítulo 4 propõe uma metodologia para descrever um cenário básico de uma rede *Ad Hoc*. Em seguida apresenta as justificativas para a escolha dos recursos de apoio para atingir os objetivos deste trabalho. Este Capítulo descreve um cenário de rede *Ad Hoc* usando tráfego VoIP como estudo de caso em que será aplicado a metodologia proposta, mostrando o *script* OTcl gerado para este cenário. O Capítulo 5 apresenta as considerações finais relativas aos assuntos tratados no texto e propostas para futuros trabalhos.

Capítulo 2

Referencial Teórico

Atualmente, a maior parte dos equipamentos de comunicação de dados trocam informações entre si utilizando uma infra-estrutura de rede cabeada (rede fixa), mas em poucos anos será comum as pessoas se comunicarem a partir de um dispositivo portátil. Este novo ambiente chamado de computação móvel, proporciona mobilidade e acesso a serviços independente de onde estão localizados. O Instituto de Engenheiros Eletrônicos e Eletricistas (IEEE) definiu o padrão 802.11 [31] para as redes sem fio que podem ser classificadas como estruturadas ou espontâneas (também conhecidas como *ad hoc*).

Na Seção 2.1 são apresentados definições, características, aplicações e os problemas enfrentados em redes *Ad Hoc*. Na Seção 2.2 são comentados as características e o funcionamento de alguns algoritmos de roteamento aplicados em redes *Ad Hoc*. Na Seção 2.3 é apresentado a ferramenta de simulação Network Simulator e uma visão geral da arquitetura desta ferramenta. Na Seção 2.4 são apresentados os propósitos de modelar sistemas e os diagramas da UML empregados na Engenharia de Software.

2.1. Redes *Ad Hoc*

A maioria dos dispositivos portáteis não se comunica diretamente entre si e uma nova arquitetura de rede chamada de rede *Ad Hoc* tem surgido para permitir que os dispositivos portáteis possam se comunicar e interagir entre si de forma espontânea, sem a necessidade de elementos centralizadores (como pontos de acesso).

O termo *Ad Hoc* implica que a rede é estabelecida para um serviço especial, por um período limitado de tempo [27]. As redes *Ad Hoc* são indicadas em situações onde não se pode ou não há necessidade de instalar uma rede fixa. Reuniões entre empresários ou alunos e

professores participando de aulas interativas através de *laptops*, em conferências, soldados em campos de batalha trocando informações estratégicas são algumas situações em que redes *Ad Hoc* podem ser usadas.

Uma rede *Ad Hoc* pode ser heterogênea, composta por diferentes tipos de dispositivos portáteis como por exemplo, PDAs e *notebooks*, ou pode ser composta somente com o mesmo tipo de dispositivo, por exemplo, um conjunto de *laptops*. Os dispositivos portáteis que estão dentro de uma mesma área são capazes de comunicarem diretamente. Caso contrário, dois dispositivos portáteis mutuamente inalcançáveis precisam se comunicar procurando um outro dispositivo que seja alcançável por ambos, caracterizando uma rede *Ad Hoc* de múltiplos saltos [3].

As dificuldades em redes *Ad Hoc* decorrem quanto há mobilidade, variações das condições de comunicação e gerenciamento de energia.

a) A mobilidade é um aspecto importante em redes *Ad Hoc*, pois a localização de um dispositivo muda constantemente de posição de acordo com seus movimentos. Dessa forma, a mobilidade implica na questão do roteamento que envolve a determinação dos caminhos e transporte de pacotes, em situações em que ocorrem mudanças na topologia. O projeto de um algoritmo de roteamento exige requisitos que leva em consideração atrasos, perdas de pacotes temporariamente, disponibilidade de canais, independência de plataforma e arquiteturas, simplicidade, robustez, mudanças de localização, convergência para melhores rotas e QoS (Qualidade de Serviço)[1]. Alguns dos mais importantes algoritmos de roteamento usados em redes *Ad Hoc* são descritos na Seção 2.2 para mostrar as características de funcionamento de cada um deles.

b) Os canais de comunicação em redes *Ad Hoc* possuem menor banda que em redes estruturadas. Taxas de erros e ruídos provocados por desconexões ao acesso à rede podem produzir variações nos canais de comunicação em redes *Ad Hoc* [20].

c) O gerenciamento de energia é outro aspecto importante nos dispositivos móveis, pois é um recurso limitado e o seu consumo deve ser minimizado. Algumas situações como, por exemplo luz de fundo de tela ou acesso a disco que está sem uso naquele instante, têm contribuído para o aumento do consumo de energia em dispositivos móveis [20].

Esses fatores têm conduzido as organizações e instituições de pesquisas a necessitarem de pessoas com maior conhecimento para planejar, projetar, implementar, operar e gerenciar sistemas nesse ambiente buscando soluções para essas dificuldades [20].

Para a modelagem de simulações de redes *Ad Hoc* no *ns-2* é importante organizar as informações que são consideradas relevantes para o experimento, representando os elementos do cenário da rede com partes bem definidas usando os diagramas. O uso da modelagem permite tratar gradualmente com a complexidade do cenário da rede *Ad Hoc* iniciando da visão mais abstrata até seu detalhamento [26].

Movimento, atualização da posição do dispositivo periodicamente, capacidade de transmitir e receber sinal por um canal de rede sem fio e dispositivos portáteis que não estão conectados por *links* são alguns aspectos que incrementam dificuldades de simulação de redes *Ad Hoc* em relação as redes convencionais.

2.2 Algoritmos de Roteamento em Redes *Ad Hoc*

Os algoritmos de roteamento em redes convencionais não são apropriados em redes *Ad Hoc*, devido as particularidades que envolvem movimento, atualização da posição constante do dispositivo, menor largura de banda e escassez de energia comentados na Seção 2.1. Os algoritmos de roteamentos DSR, TORA, AODV e DSDV são brevemente comentados mostrando o funcionamento de cada um deles.

a) Algoritmo DSR (Dynamic Source Routing) [2]

O DSR é um algoritmo em que o nodo de origem determina toda a seqüência de nodos por onde passará o pacote até chegar ao seu destino. Os cabeçalhos dos pacotes de

dados contêm informações desta seqüência de nodos. Quando um nodo precisa enviar um pacote para outro, o nodo de origem verifica se possui uma rota para o de destino em seu *cache*. Caso a rota não exista, inicia um processo de descoberta de rotas nas tabelas para encontrar dinamicamente uma rota para o destino. Nesse processo de descoberta, o nodo de origem pode detectar a ocorrência de alterações na topologia da rede que poderão compromenter o uso das rotas. Por exemplo, se um nodo identifica problema de comunicação, o nodo vizinho envia um pacote do tipo ROUTE ERROR para o nodo de origem.

b) Algoritmo TORA (Temporally-Ordered Routing) [28]

O TORA é um algoritmo robusto e foi projetado para diminuir as reações do protocolo às mudanças de topologia [1]. Este protocolo trabalha de forma independente em cada nodo. Cada nodo tem informações referentes apenas a seus vizinhos. Quando um nodo precisa de uma rota para um destino específico, o nodo transmite um pacote de requisição, o qual contém o endereço do destino da rota.

c) Algoritmo AODV (*Ad Hoc* On-Demand Distance Vector) [29]

O algoritmo AODV é baseado no algoritmo DSDV, porém, o AODV tenta eliminar a necessidade de um *broadcast* global das mensagens que são disparadas ou requisitadas [1]. No AODV, o nodo que deseja conhecer uma rota para um destino específico manda mensagens do tipo ROUTE REQUEST. Esta mensagem é encaminhada pelos nodos intermediários que aproveitam para atualizar as suas tabelas de roteamento. Quando a mensagem chega ao destino, uma resposta do tipo ROUTE REPLY é produzida e encaminhada ao nodo de origem. A resposta do tipo ROUTE REPLY contém a quantidade de nodos necessários para atingir o destino.

d) Algoritmo DSDV (Destination-Sequence Distance Vector) [30]

No DSDV, as mensagens de roteamento são trocadas entre os nodos vizinhos dentro de uma área de cobertura. As mensagens de atualização de rotas também são disparadas quando ocorrem mudanças nas tabelas de roteamento dos nodos. Além disso, as mensagens de requisição de rota também podem ser disparadas quando a rota para um destino não é

conhecida. Neste caso, os pacotes são armazenados em um *cache* até que receba uma resposta a uma mensagem de requisição de rota feita anteriormente.

Em [3] foi realizado um estudo comparando os algoritmos AODV, DSDV e DSR. O algoritmo DSDV apresenta dificuldades para manter as rotas válidas, como conseqüência, muitos pacotes são perdidos. O tráfego na rede aumenta dependendo da mobilidade dos nodos para manter rotas para todos os outros nodos. No DSR, a fonte captura várias rotas alternativas para o mesmo destino, porém, em rotas que envolvem muitos saltos acarreta aumento na sobrecarga de roteamento em *bytes*. No AODV, é mantido uma tabela com uma entrada para cada destino, apresentando uma baixa sobrecarga de roteamento em *bytes*, pois os pacotes de dados carregam apenas o endereço de destino e não as rotas para alcançar o destino.

Em [1] comenta que o TORA é indicado para uso em redes móveis densas. Este algoritmo teve o pior desempenho em relação aos outros algoritmos citados. Em redes com 10 ou 20 nodos, o algoritmo consegue entregar 90% dos pacotes gerados, porém em redes com mais de 30 nodos não consegue manipular o tráfego gerado e uma quantidade de pacotes de dados deixou de ser entregue. O estudo realizado em [34] mostrou que o TORA tem características ruins em relação ao item atraso e a reconstrução da rota pode não ocorrer rapidamente.

A partir de trabalhos relacionados [1,3,34] mostraram que o algoritmo AODV tem melhor desempenho em relação aos outros algoritmos DSDV, DSR e TORA. O AODV é uma melhoria dos algoritmos DSDV e DSR.

2.3 Ferramenta de simulação Network Simulator

A simulação tornou-se uma tarefa importante em projetos que envolvem a integração dos meios de comunicação na área de redes de computadores. O Laboratório Laurence Berkeley National desenvolveu um simulador chamado Network Simulator (*ns-2*), projetado para simulações e estudos de protocolos de redes, comportamento de tráfego, mecanismo de

gerenciamento de fila, algoritmos de roteamento, implementação de simulação multicasting e protocolos da camada MAC para redes LAN. O código fonte da distribuição do *ns-2* também oferece exemplos de implementações de protocolos de roteamento para redes *Ad Hoc* tais como AODV, TORA, DSR e DSDV para citar alguns.

O *ns-2* foi escrito na linguagem C++, pois a manipulação de *bytes* e cabeçalhos de pacotes nessa linguagem, permite a construção de algoritmos que tratem grandes conjuntos de dados, proporcionando uma velocidade maior na implementação detalhada de protocolos [12]. Além disso, o *ns-2* faz uso da linguagem OTcl (uma extensão orientada a objeto para linguagem Tcl) [17] que é interpretada e interativa, compatível com várias plataformas e proporciona interface gráfica.

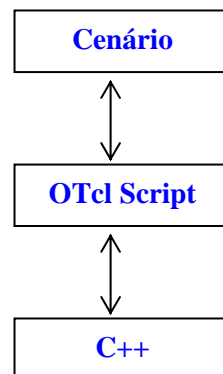



Figura 2.1: Níveis de abstração de um ambiente de simulação no *ns-2*

A Figura 2.1 mostra os níveis de abstração de um ambiente de simulação no *ns-2*. O usuário tem um cenário de rede *Ad Hoc* para ser simulado. O cenário contém vários elementos que são definições da topologia da rede, os agentes usados na camada de transporte, os protocolos usados na camada de aplicação, o tempo em que começa e pára a transmissão de pacotes e os protocolos de roteamento para redes *Ad Hoc*. O usuário escreve esses elementos em um *script* OTcl na sintaxe do *ns-2*. O *script* é um programa e ao executá-lo, os elementos são criados como objetos no interpretador OTcl. Através dos métodos definidos na classe TclClass, este interpretador vincula automaticamente cada objeto a um objeto correspondente

em C++ onde é compilado. Sendo assim, no *ns-2* existe um objeto dentro do interpretador OTcl similar a um objeto em C++.

A Figura 2.2. mostra uma hierarquia de classes em C++ e uma hierarquia de classes em OTcl. O *ns-2* separa a implementação do *script* OTcl e a implementação em C++, tratando assim de duas hierarquias semelhantes e bastante relacionadas uma com outra, uma hierarquia de classes compiladas em C++ e outra hierarquia de classes do interpretador OTcl. Caso o usuário necessite adicionar novos elementos no *ns-2*, ele precisa criar a classe do objeto na hierarquia em C++ e também na hierarquia do interpretador OTcl.

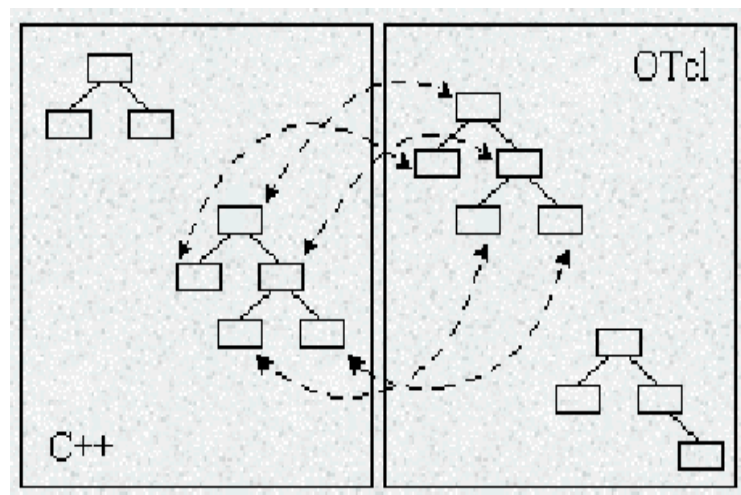



Figura 2.2: Classe em C++ e classe em OTcl [7]

O *ns-2* possui muitas classes que na maioria são componentes de rede. A Figura 2.3 mostra uma hierarquia parcial de classes em OTcl e em C++ do Network Simulator. A raiz da hierarquia é a classe TclObject que é a super classe de todos os objetos da biblioteca das hierarquias OTcl e C++. Todo objeto na classe TclObject é criado pelo usuário dentro do interpretador e um outro objeto similar é criado na hierarquia de classes em C++ a partir dos procedimentos chamados pelo interpretador.

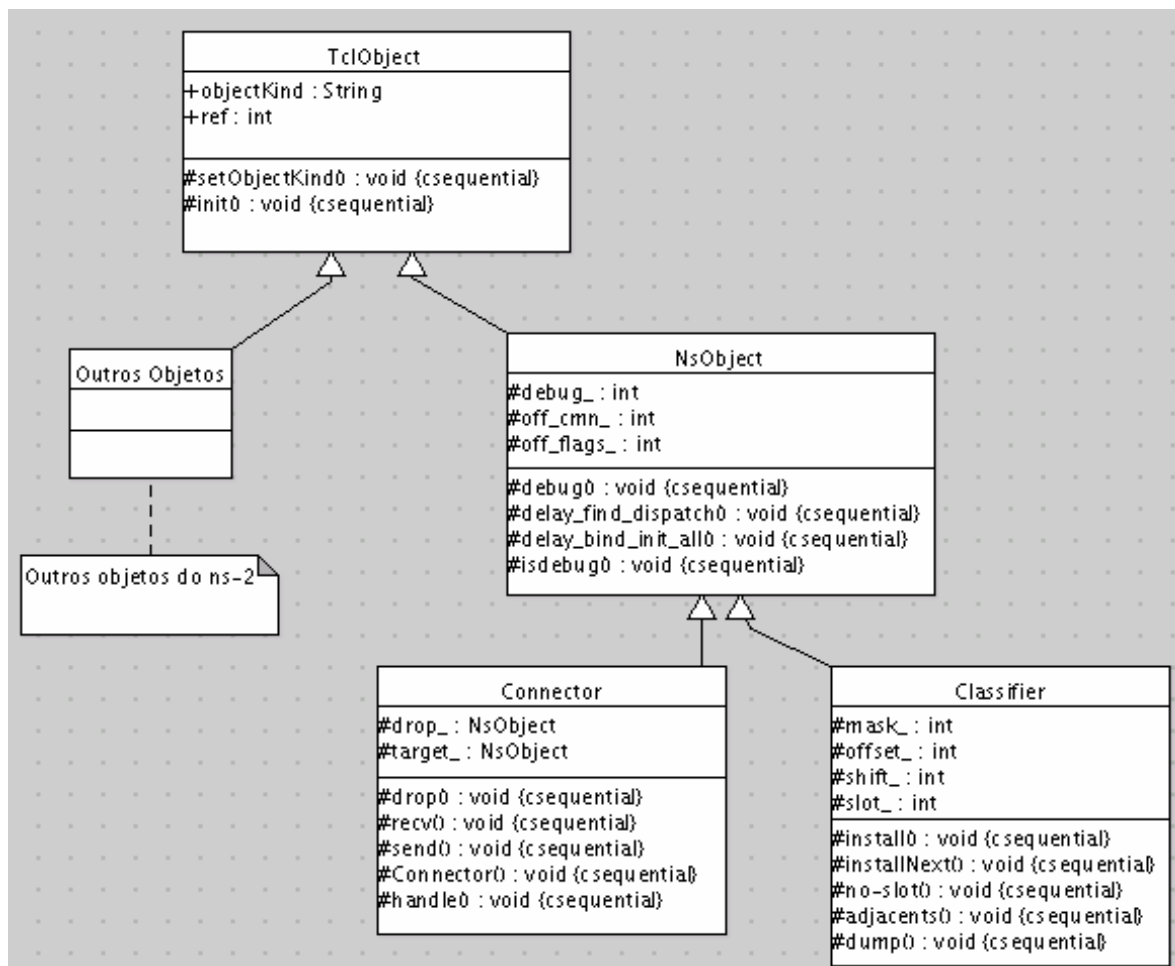



Figura 2.3 : Hierarquia parcial de classes para classe TclObject [37]

A classe NsObject é a super classe de todos os objetos de componentes de redes que envolve pacotes, nodos e *links*. A classe NsObject é generalizada por duas subclasses: Connector e Classifier.

Os objetos da subclasse Connector tem a função de gerar dados apenas a um destinatário. O pacote é entregue a um vizinho ou o pacote é descartado. Um objeto connector receberá pacote, executará alguma função como exemplo modelar características de atrasos e da banda do *link*, entregará esse pacote aos seus vizinhos ou descartará o pacote. A subclasse Classifier proporciona uma forma de mapear os valores para uma interface de saída de um objeto que é o próximo destinatário do pacote.

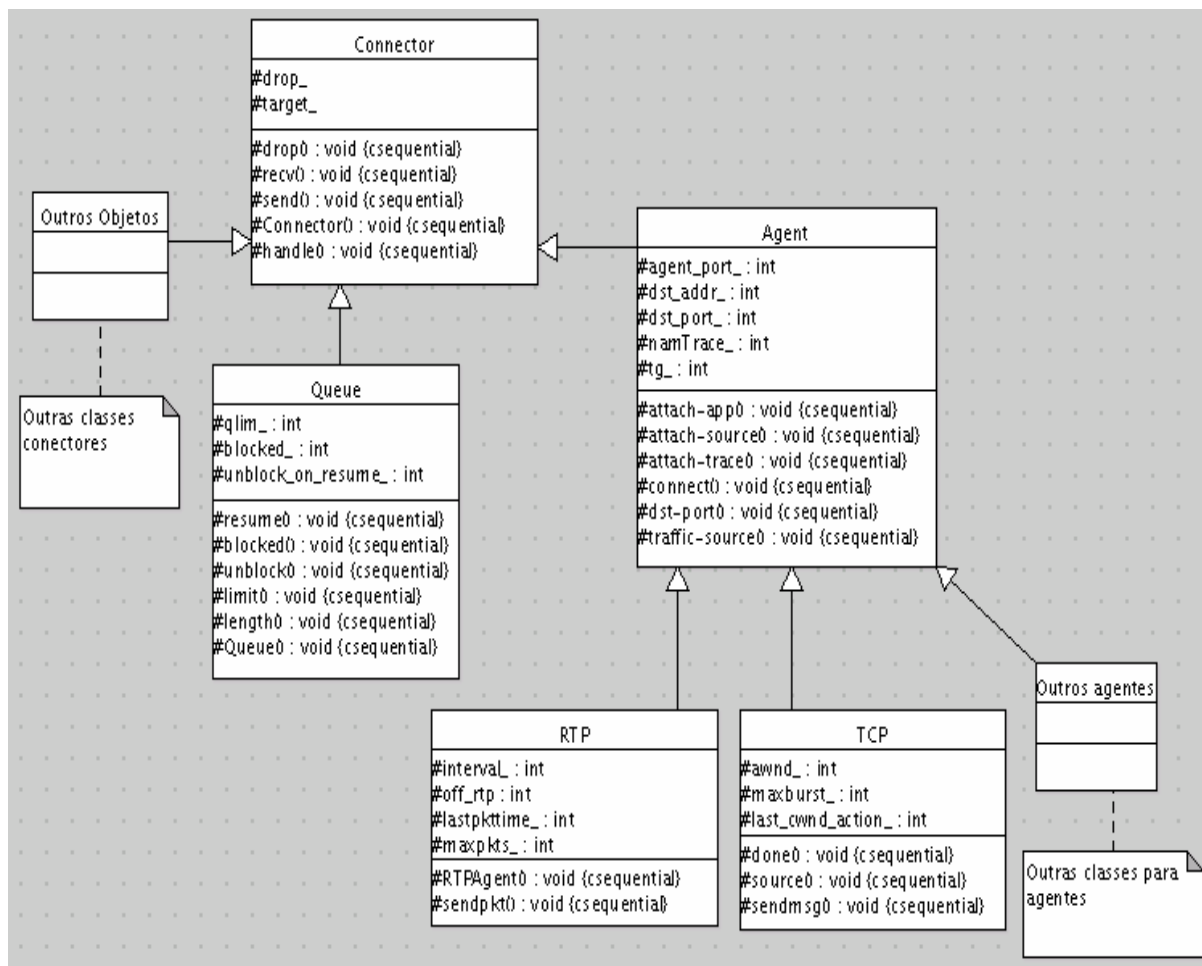



Figura 2.4:Hierarquia parcial de classes para a classe Connector [37]

A Figura 2.4 mostra a hierarquia parcial das subclasses da classe Connector. As subclasses Agent e Queue são algumas relacionadas a superclasse Connector. As classes Delay e Trace são outras classes que também herdam algumas características dessa superclasse.

A subclasse Agent é generalizada em outras classes mais especializadas. Como exemplo, a classe Agent pode ser especializada em classes TCP e RTP. O *ns-2* também simula outros tipos de agentes como UDP, NULL e LossMonitor [17].

A subclasse Queue é generalizada em outras classes mais especializadas. Como exemplo, a classe Queue pode ser especializada em classes DropTail e RED (Random Early Discard)[17]. O DropTail é uma fila do tipo FIFO (First In First Out).

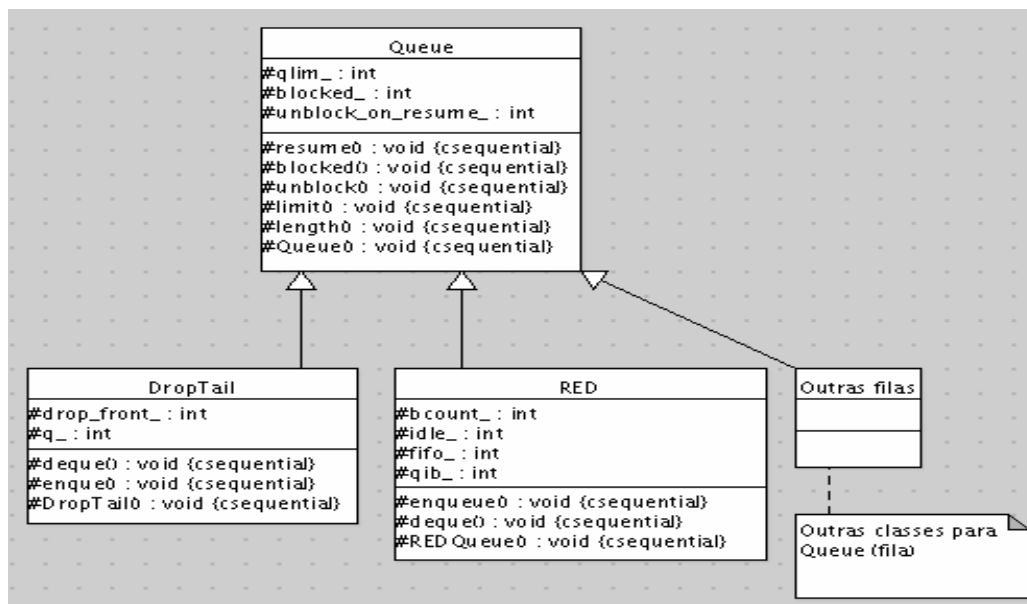



Figura 2.5: Hierarquia parcial de classes da classe Queue [37]

A Figura 2.5 mostra a hierarquia parcial de classes especializadas para a classe Queue. A classe Queue tem a função de controlar bloqueio, desbloqueio, tamanho e limite de uma fila. O *ns-2* simula outros tipos de filas como SFQ (Stochastic Fair Queueing) e PriQueue [17].

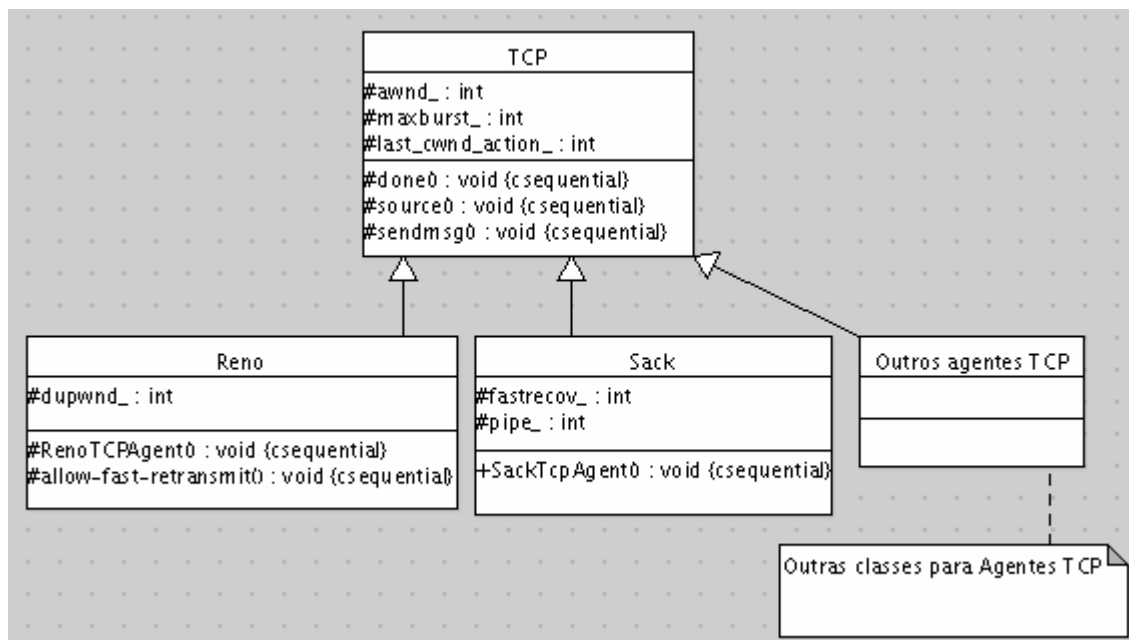



Figura 2.6: Hierarquia parcial de classes da classe TCP [37]

A Figura 2.6 mostra a hierarquia parcial das classes para a subclasse TCP que pode ser especializada em Reno e Sack. Outros tipos de classes especializadas para TCP são TCPSink, NewReno, Sack1 e Vegas.

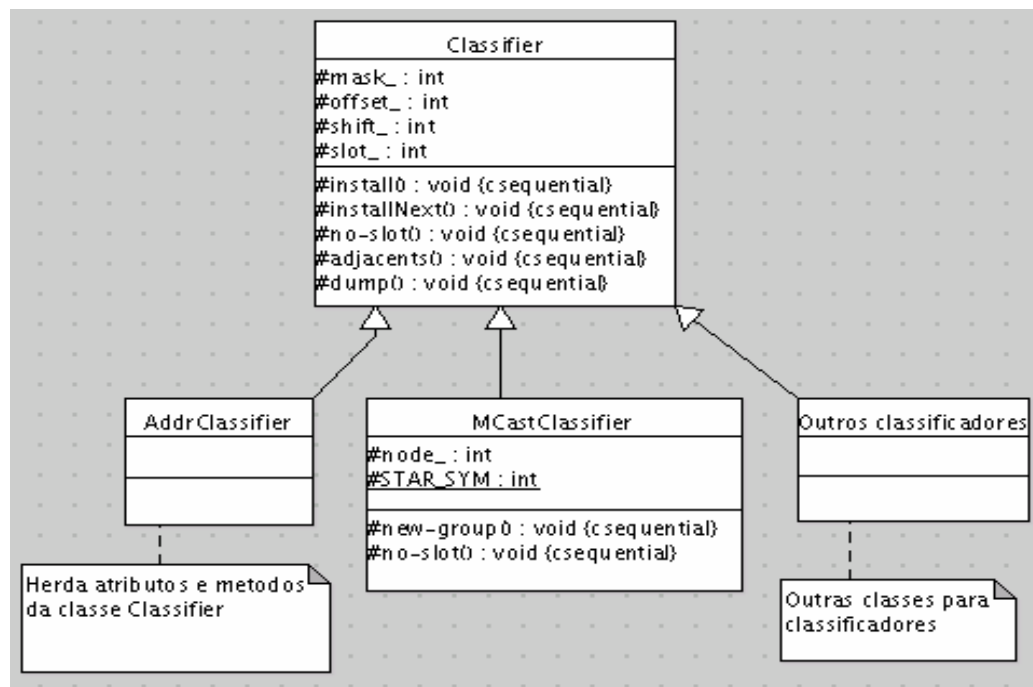



Figura 2.7: Hierarquia parcial de classes para classe Classifier [37]

A Figura 2.7 mostra a generalização da classe Classifier para outras subclasses: AddrClassifier e McastClassifier. A subclasse AddrClassifier é um classificador que suporta roteamento unicast, onde o emissor cria um canal de comunicação separado para cada destinatário. A subclasse McastClassifier é um classificador utilizado na transmissão de pacotes de acordo com os endereços dos grupos de origem e destino, através de uma Tabela de mapeamento que contém os grupos e os seus respectivos números gerados por este classificador. Nessa hierarquia existem outros tipos de classificadores tais como HashClassifier e Replicator.

Outras classes existentes no código fonte do *ns-2* são Tcl, TclClass, TclCommand, EmbeddedTcl e Scheduler. A classe Tcl contém procedimentos que o código C++ usará para

acessar e comunicar com o interpretador OTcl. A classe TclClass tem duas funções: construir um vínculo entre a hierarquia das classes do interpretador OTcl com a hierarquia das classes em C++ e proporcionar procedimentos para construir novos objetos pela classe TclObject. A classe TclCommand proporciona um mecanismo para o *ns-2* transferir comandos simples para interpretador e que este possa executá-los dentro do contexto global. A classe EmbeddedTcl contém procedimentos responsáveis pela construção dos comandos usados no programa OTcl. Scheduler é uma classe implementada em C++ que tem a função de disparar os eventos existentes no decorrer de uma simulação e acionar o objeto que realizará o tratamento do evento.

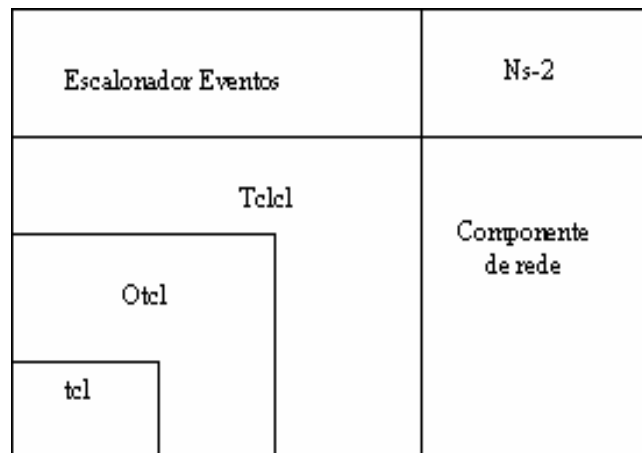



Figura 2.8 : Arquitetura do Network Simulator [7]

A Figura 2.8 apresenta a arquitetura do *ns-2*. A camada tcl é onde o usuário atua escrevendo e simulando os *scripts* usando os objetos na biblioteca OTcl. O escalonador de eventos e os componentes de redes são, na maioria, implementados em C++, que estão disponíveis ao interpretador OTcl através dos procedimentos implementados na camada Tclcl.

Comunidades de pesquisas têm usado essa ferramenta, desenvolvendo e adicionando seus próprios módulos de acordo com suas necessidades. Alguns módulos encontram-se disponíveis dentro do *ns-2* como por exemplo programas desenvolvidos pelo grupo

Monarch[1]. Esses programas geram separadamente *scripts* OTcl contendo a criação e movimentos dos nodos móveis e padrões de conexões de tráfegos para serem utilizadas em simulações cujas topologias são mais extensas. Usuários podem criar programas específicos para simular determinados tipos de problemas como por exemplo gerador que simula tráfego de voz via IP (VoIP). Nesse contexto, C-N. Chuah [24] desenvolveu um programa que simula a geração de voz usando um modelo Markoviano, variáveis aleatórias de distribuição uniforme e exponential para tráfego CBR. O usuário pode usá-lo em seu *script* OTcl fazendo alguns ajustes necessários correspondentes ao seu cenário para simular tráfego VoIP sobre redes *Ad Hoc*. O *ns-2* pode ser obtido no site http://www.isi.edu/nsnam/ns, assim como o manual de documentação oficial.

2.4 Engenharia de Software

Com um mercado mais competitivo, as mudanças são constantes e cada vez mais necessário administrar aplicativos de alta qualidade. A Engenharia de Software apresenta um papel importante para o processo de desenvolvimento de um software, abrangendo um conjunto de etapas que envolvem métodos, ferramentas e procedimentos. Essas etapas proporcionam os detalhes de como planejar, analisar requisitos de sistemas, projeto da estrutura de dados, arquitetura de programa e algoritmo, para auxiliar a concepção e a construção de softwares com índices maiores de qualidade e produtividade.

A Engenharia de Software emprega métodos e técnicas para representar modelos do mundo real considerando os pontos principais para facilitar o seu entendimento. Existem várias propostas de métodos e técnicas, e atualmente, a que mais se destaca é o da orientação a objetos[14]. A UML é uma linguagem visual utilizada para modelar sistemas computacionais por meio do paradigma de Orientação a Objetos. Em 1997 a UML foi aprovado pelo OMG

1 Monarch (Mobile Networking Architecture) é um grupo direcionado a projetos quanto ao desenvolvimento de protocolos e interfaces de protocolos na área de redes sem fio. Redes ad hoc têm sido um dos interesses deste grupo.

(Object Management Group) [19] como o padrão adotado pela maior parte do mercado na atualidade.

Modelar um sistema é uma tarefa extensiva, sendo necessária a descrição de vários aspectos tais como a estrutura estática e interação dinâmica, e também a parte organizacional. O modo de descrever esses aspectos de modelagem pela UML é através da notação definida pelos seus vários diagramas. Um diagrama é uma notação gráfica de um conjunto de elementos de modelo, conectado de arcos (relacionamentos) e vértices (outros elementos do modelo). Cada diagrama representa uma visão enfatizando um aspecto particular do sistema ou de uma organização [14]. Os diagramas mais conhecidos e usados na UML são diagrama de caso de uso, diagrama de classe, diagrama de atividade, diagrama de componentes e diagrama de distribuição.

O *ns-2* trata de vários elementos de redes *Ad Hoc*. Esses elementos podem ser representados por classes. Os eventos para acionar o comportamento de algum elemento da rede podem ser representados através de diagramas de atividades. As bibliotecas e códigos executáveis do *ns-2*, por exemplo, podem ser representados por diagramas de componentes. Dessa forma, cada diagrama UML analisa parte da rede *Ad Hoc*, especificando e documentando o processo da simulação de um cenário de uma rede.

2.4.1 Diagrama de Caso de Uso

O entendimento de todos os elementos existentes em uma rede de computadores é uma tarefa difícil e importante para desenvolver um sistema que atenda as suas necessidades. O caso de uso é a principal técnica utilizada na fase de levantamento de necessidades que envolve os objetivos, regras, recursos, ações, as restrições do sistema, representando uma visão mais de alto nível de funcionalidade. Os diagramas de caso de uso são fáceis de entender e podem ser úteis por analistas de redes e pesquisadores na área de redes, e assim tornam uma excelente ponte entre quem desenvolve o sistema e os utilizadores finais.

O diagrama de caso de uso tem propósitos de:

a) descrever os requisitos funcionais do sistema entre as pessoas envolvidas no negócio;
b) fornecer uma descrição sobre as responsabilidades que devem ser tratadas pelo sistema;
c) oferecer situações do mundo real para o teste do sistema.

O diagrama de caso de uso tem três elementos básicos: ator, caso de uso e interação. A seguir é descrita a definição e aplicação de cada um dos elementos.

Ator - um ator é um agente que interage com o elemento externo do domínio do problema. O ator pode ser uma instituição, organização ou categoria com papel definido, que pode ser uma pessoa, máquina ou dispositivo. Um ator pode ser um sistema externo que precisa de informações acerca do sistema que está sendo modelado [14]. Na área de redes, o ator pode ser um outro sistema de redes que tem interação com o sistema de redes que está sendo modelado. Em redes *Ad Hoc*, um ator pode ser designado para representar pessoas compartilhando informações em reunião ou participando de aula interativa usando laptops, ou soldados num campo de batalha trocando informações estratégicas [1].

Caso de uso – um caso de uso é um comportamento de um sistema, que serve para especificar as suas necessidades e funcionalidades. O caso de uso especifica uma sequência de ações de interação mostrando como os eventos externos são respondidos [14].

Interação – uma interação no diagrama de caso de uso representa um fluxo indicando a comunicação do ator com um sistema. Além disso, esse fluxo serve para mostrar comportamento de exceção e casos especiais entre casos de uso. Trata-se do relacionamento de caso de uso para outro, especificando como o comportamento definido para o primeiro caso de uso pode ser inserido no comportamento definido para o segundo caso de uso [14].

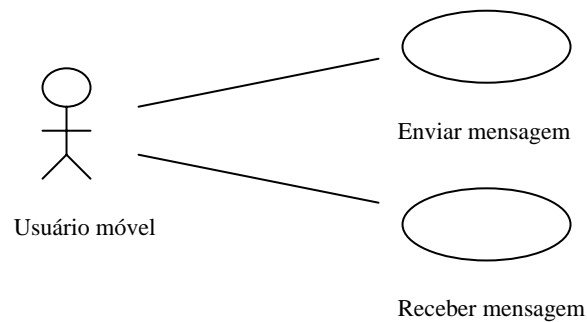



Figura 2.9: Diagrama de caso de uso

A Figura 2.9 mostra um diagrama de caso de uso, onde o ator é representado por um usuário móvel que tem como ações enviar mensagem e receber mensagem. As ações são representados por uma elipse denominado casos de uso. A interação entre o ator e os casos de usos são representados por uma linha.

2.4.2 Diagrama de classes

A modelagem de um sistema envolve a identificação de elementos importantes de acordo com o domínio do problema. Esses elementos podem constituir de algumas propriedades semelhantes. Em UML, esses elementos são chamados de objetos. Os objetos incluem abstrações que fazem partem do sistema e formarão uma distribuição de responsabilidades em um sistema. Em UML, todos esses objetos são modelados como classes. Classe é a representação de informações e comportamento que compartilham as mesmas características e semânticas. Uma classe é um grupo de objetos relacionados, mas não necessariamente idênticos.

Dois objetos podem ter as mesmas características, porém, podem ter valores diferentes armazenados através de um conjunto de operações. Um objeto representa uma ocorrência de uma classe. Um atributo é a menor unidade de informação associada a uma classe, ou seja,

do armazenamento de valores em células. Uma operação é um comportamento associado a uma classe, resultante de um procedimento algorítmico.

A Figura 2.10 mostra uma classe. Cada classe deve ter um nome que a diferencie de outras classes. Os atributos representam propriedades para um objeto da classe Nodo, que estão listados em um compartimento abaixo do nome da classe. As operações representam ações que podem ser solicitadas por um objeto da classe Nodo para alterar algumas de suas propriedades. As operações devem ser listadas em um compartimento abaixo dos atributos da classe.

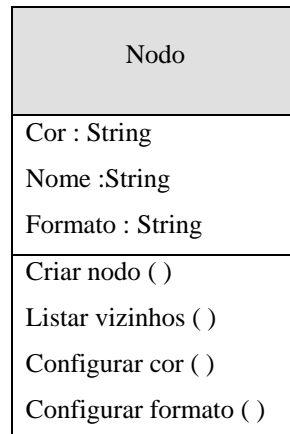



Figura 2.10: Notação da classe.

O diagrama de classe é a essência da UML e trata-se de um conjunto de classes denotando a estrutura estática do sistema. A modelagem usando o diagrama de classe pode auxiliar nas decisões de quais as abstrações de classes fazem parte do sistema e quais estão fora dos limites do sistema. Os diagramas de classes são importantes não só para a visualização, a especificação e a documentação de modelos estruturais, mas também para a construção de sistemas executáveis por meio de engenharia reversa.

A Figura 2.11 mostra um diagrama de classes com alguns elementos usados em uma rede *Ad Hoc*. O diagrama contém quatro classes: topologia, nodo_móvel, tráfego e protocolo. Uma topologia consiste de nodos móveis. Uma rede A*d Hoc* adota um protocolo de roteamento

que estabelece as rotas entre os nodos móveis. As rotas entre nodos móveis gera um tráfego na rede A*d Hoc*.

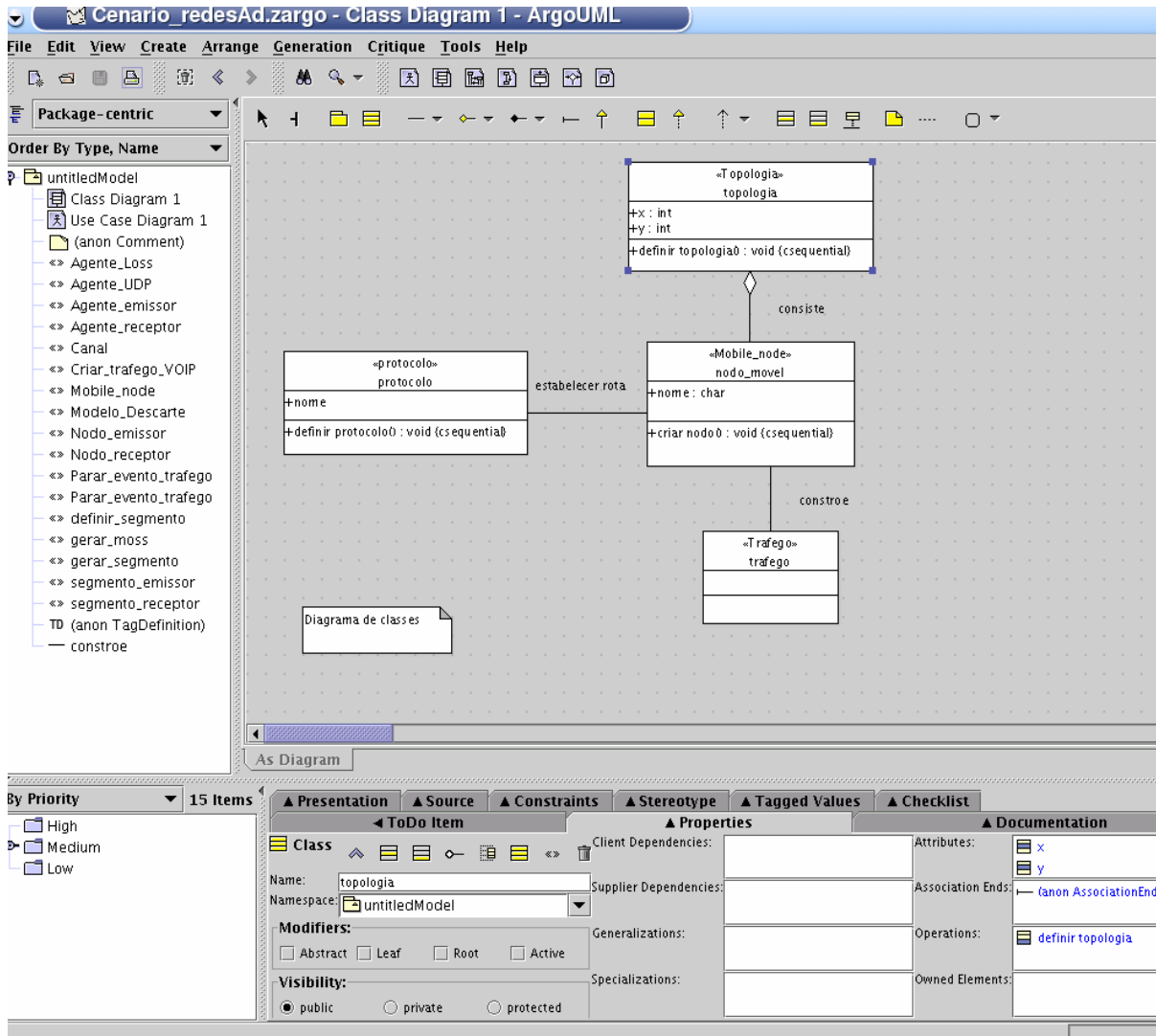



Figura 2.11: Diagrama de classes

Outro aspecto importante, é a representação de como os objetos estão relacionados uns com os outros. Os relacionamentos podem ser complexos, por isso, a UML fornece outras notações que permitem a visualização, especificação, construção e documentação deles em

qualquer nível de detalhe da modelagem do sistema. Os principais relacionamentos usados no diagrama de classe são agregação, composição, associação e generalização/especialização.

Agregação é uma forma de representar que um objeto é constituído pelo menos de uma parte em relação ao todo. O objeto agregado pode existir sem os seus objetos constituintes. Este tipo de relacionamento significa que um objeto do todo contém os objetos das partes. O símbolo de um pequeno diamante aberto aparece na extremidade do objeto agregado como mostra a Figura 2.12.

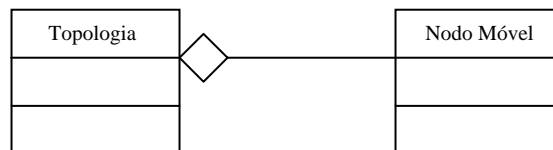



Figura 2.12: Notação de agregação

A Figura 2.12 mostra uma estrutura todo-parte entre as classes Topologia e Nodo Móvel. A topologia consiste de nodos móveis. Uma topologia pode existir sem ter nodos móveis.

Composição é um relacionamento em que o objeto todo constitue de objetos partes. O que diferencia da agregação é que em um relacionamento por composição o objeto composto não pode existir sem os seus objetos constituintes. O símbolo de um pequeno diamante preto aparece na extremidade do objeto composto como mostra a Figura 2.13.

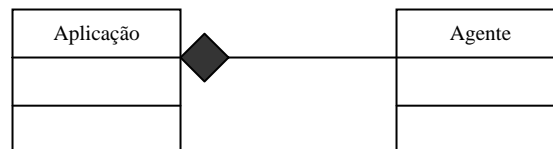



Figura 2.13: Notação de composição

A Figura 2.13 mostra uma estrutura todo-parte representando uma composição entre as classes Aplicação e Agente. Um serviço utilizado pela camada de aplicação necessita de agente na camada de transporte para criar um canal de comunicação na rede.

Associação descreve um vínculo entre duas classes, determinando que as instâncias de uma classe estão ligadas às instâncias da outra classe. Este relacionamento pode envolver troca de informações entre as classes ou compartilhamento de métodos. A associação pode ainda identificar algum nível de dependência entre as classes que a compõe [21]. Uma associação é representada por uma linha sólida, conectando a mesma classe ou classes diferentes como mostra a Figura 2.14.

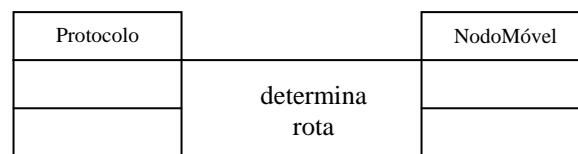



Figura 2.14: Notação de associação

A Figura 2.14 mostra que nodo móvel determina rota através de um protocolo de roteamento.

Generalização é um relacionamento entre uma classe mais geral, chamado de superclasse, e uma classe mais específica chamado de subclasse, que é a especialização. Este relacionamento é usado para descrever todos atributos e operações comuns de um grupo de objetos formando uma classe mais genérica e descrever os atributos e operações para aqueles objetos que são específicos formando subclasses da superclasse. O propósito mais conhecido para este tipo de relacionamento é compartilhar os atributos e operações da superclasse, ou seja, os objetos da subclasse herdam os atributos definidos na superclasse. Esse mecanismo é chamado de herança. A generalização/especialização é representada por um triângulo conectado na superclasse, estendendo-se por um linha reta até a subclasse [35].

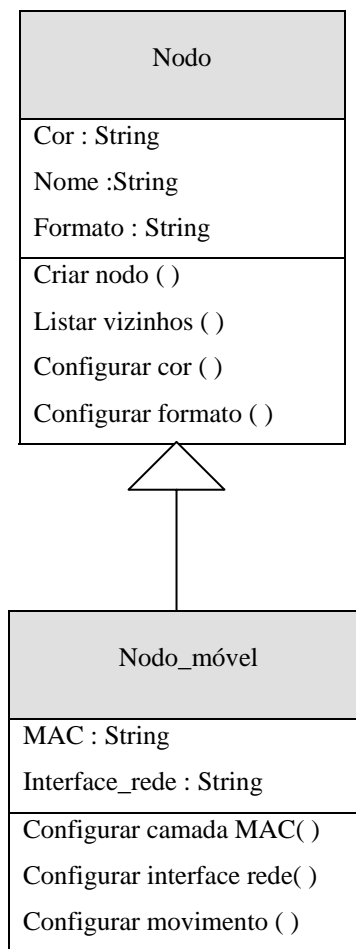



Figura 2.15: Notação de generalização/especialização

A Figura 2.15 mostra uma generalização/especialização para o elemento nodo. O nodo móvel é um nodo, herdando atributos e operações da superclasse Nodo. A subclasse Nodo_móvel contém atributos e operações adicionais que caracterizam a mobilidade do nodo.

2.3.3 Diagrama de atividades

O objeto é uma instância da classe. A instância pode se apresentar em um ou mais estados. A UML trata a representação do comportamento desses objetos através do diagrama de atividade. Um diagrama de atividades mostra o fluxo de uma atividade para outra. Uma atividade é um tipo de comportamento que um objeto carrega enquanto estiver em um

determinado estado. Esta atividade pode ser interrompida, ou ser executada ao mesmo tempo em que o objeto está em um determinado estado, ou ser interrompida pelo objeto que está passando para um outro estado. As atividades resultam em alguma ação, formada por um procedimento que gera uma mudança de estado do sistema ou o retorno de um valor. Dessa forma, o diagrama de atividade permite escolher a ordem pela qual as tarefas devem ser feitas, mostrando como o objeto muda de um estado para outro e como o objeto se comporta de forma diferente em cada um dos estados [16].

Enquanto os diagramas de classes são usados para representar os aspectos estáticos, os diagramas de atividades são empregados para representar os aspectos dinâmicos do sistema. Com os diagramas de atividades, é possível preparar roteiros de cenários, envolvendo a interação de objetos de interesse e as mensagens que poderão ser trocadas entre eles. Uma classe pode ter um ou mais diagramas de atividades. Para representar o comportamento dos objetos são usados cinco tipo de informações: estado de ação, fluxo de objetos, transições, ramificações e sincronização.

Estados de ação

Um estado de ação representa a realização de uma ação dentro de um fluxo de controle, ou seja, não pode ser decomposto em sub-estados. Geralmente, o uso de estado de ação na modelagem é para mostrar um passo na execução de um algoritmo. Uma atividade pode ter vários estados de ação. Um estado de ação é representado por um retângulo com bordas arredondadas sem divisões.

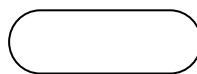



Figura 2.16: Estado de ação

O início e o término da sequência de vários estados de ação são representados com um pseudo estado inicial desenhado como um círculo sólido e um pseudo estado final é desenhado com um círculo que cerca um círculo sólido pequeno como mostra a Figura 2.17.

Figura 2.17: Estado inicial e estado final

Fluxo de objetos

O fluxo de objetos descreve o estado dos objetos envolvidos na atividade descrita no diagrama. O estado do objeto é representado pelo valor em seu atributos. Um fluxo de objeto é representado por uma reta tracejada contendo uma seta que atinge um quadrado representando um objeto de uma determinada classe [21]. A Figura 2.18 mostra que no estado de ação *Criar nodo*, o objeto naquele momento, tem as características como o atributo *nome* igual a node_(2) e *cor* igual azul.

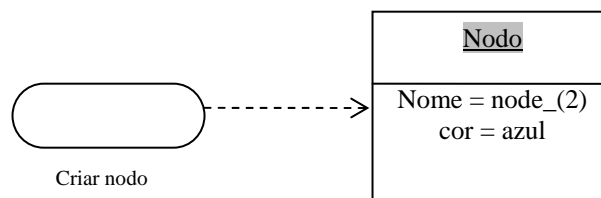



Figura 2.18: Fluxo de objeto da classe Nodo

Transição

Quando uma atividade contém vários estados de ação, é necessário representar como o fluxo ocorre entre os estados de ação. Essa representação é chamada de transição. A transição mostra o caminho de um estado de ação para outro, especificando que um determinado estado de ação quando está completo, passa para o estado seguinte de ação. A transição é

representada por uma linha simples com uma direção. A Figura 2.19 mostra um estado de ação *Criar nodo* com transição para o estado de ação *Criar movimento nodo*.

Figura 2.19: Transição em dois estados de ação

Sincronização

Transições sequenciais e ramificações são caminhos mais comuns de serem usados em diagrama de atividade, porém é possível encontrar fluxos concorrentes de estados de ação. A sincronização é uma forma de mostrar que dois ou mais estados de ação ocorrem em paralelo como mostra a Figura 2.20.

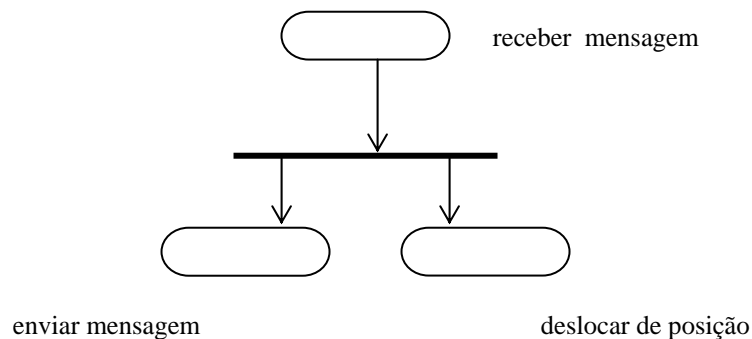



Figura 2.20: Sincronização entre estados de ação

A Figura 2.20 mostra a representação um elemento móvel recebendo uma mensagem [18]. O elemento móvel toma uma ação de enviar uma mensagem e ao mesmo tempo, movimentar-se em uma topologia de uma rede *Ad Hoc*. Os estados de ação *enviar mensagem* e *deslocar de posição* são representados como fluxos concorrentes após a barra de sincronização.

Ramificação

As transições sequenciais simples não são o único tipo de caminho para ser representado em um fluxo de controle, é possível incluir caminhos alternativos para os estados de ação. Os caminhos alternativos são mostrados como ramificação. A ramificação é representada como um diamante.

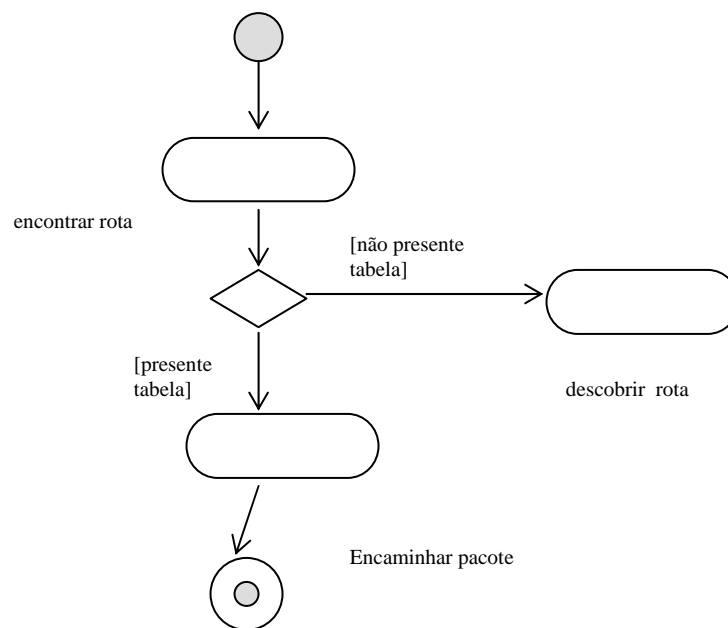



Figura 2.21: Ramificação entre estados de ação

A Figura 2.21 mostra uma ramificação que determina transições com condições para estados de ação diferentes, considerando uma rede *Ad Hoc* que utiliza um protocolo de roteamento AODV onde um elemento móvel necessita encontrar uma rota (estado de ação *encontrar rota*) para entregar pacote. Os caminhos alternativos que geram uma ramificação podem ser a presença ou não presença da rota em uma tabela de roteamento. Quando a rota não se encontra na tabela, inicia um processo de descoberta da rota, representado pelo estado de ação *descobrir rota*. Caso contrário, o pacote é encaminhado pela rota disponível na tabela.

2.3.4 Diagramas de implementação

A UML proporciona modelagem de uma arquitetura descrevendo a decomposição detalhada do hardware e software que cercam a implementação de um sistema. Nesta etapa, é desenhado um mapeamento da arquitetura lógica de classes para uma arquitetura física em termos de componentes e módulos de códigos que implementam o conceito definido para o sistema. Os diagramas de implementação são apresentados sob duas formas na UML como o diagrama de componentes e o diagrama de implantação.

Diagrama de componentes

Um componente representa uma unidade de código de software (fonte, binário ou executável). O diagrama de componente é empregado para a modelagem da visão estática de implementação de um sistema, configurando todos os arquivos de códigos-fonte e a tecnologia usados para o desenvolvimento do sistema, que podem auxiliar no gerenciamento da configuração e no controle de versão. Este diagrama mostra as dependências entre componentes de sistema, inclusive componentes de códigos.

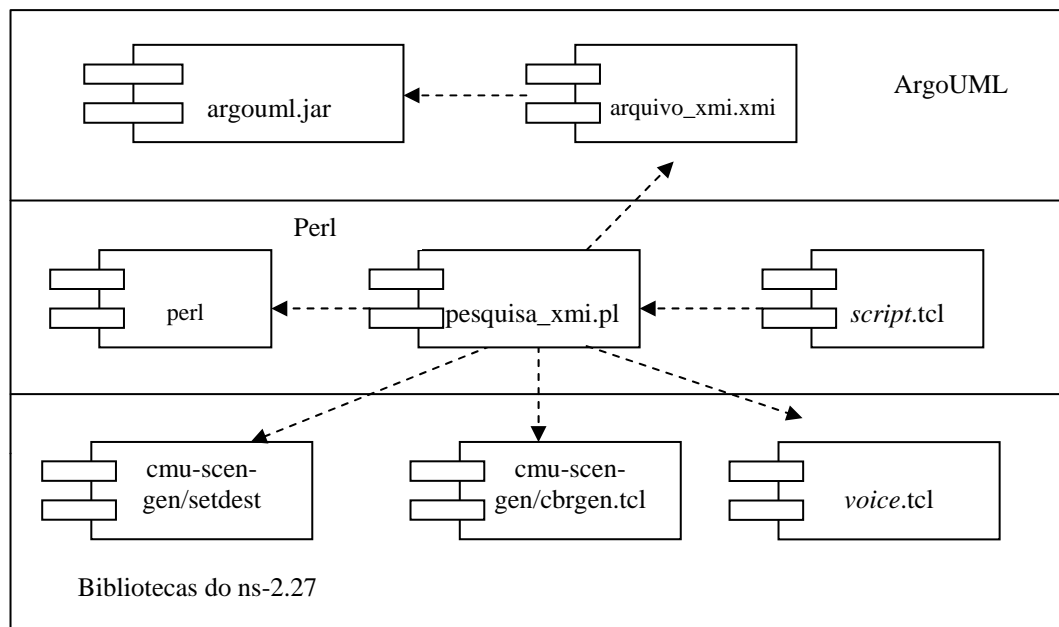



Figura 2.22: Diagrama de componentes

A Figura 2.22 mostra no primeiro bloco um componente executável denominado *argouml.jar*. Este componente é a ferramenta ArgoUML que produzirá um arquivo com extensão xmi, representando por um componente denominado *arquivo_xmi.xmi*. O segundo bloco mostra uma unidade de código executável chamado de *pesquisa_xmi* implementado na linguagem perl que será usado para ler um arquivo xmi que contém a representação de um cenário de redes em diagramas UML. Além disso, o código executável *pesquisa_xmi* tem dependência de duas bibliotecas do *ns-2* que estão representadas no terceiro bloco, e tem dependência do executável (*voice.tcl*) que é um *script* OTcl para simulação de tráfego VoIP. O componente *script.tcl* é um arquivo que será gerado com a sintaxe em *ns-2* do cenário representado em UML pelo componente *pesquisa_xmi.pl*.

Diagrama de implantação

A UML fornece o diagrama de implantação para mostrar a organização do hardware e a ligação do sistema aos dispositivos. Um dispositivo é um elemento físico que existe em tempo de execução e representa um recurso computacional. Geralmente, o dispositivo é determinado por um estereótipo como processador, impressora, disco, cliente/servidor e outros. Este diagrama auxilia a visualizar ou especificar a distribuição física dos componentes e dispositivos que compõem o sistema, os processos e objetos que neles se mantêm. Dessa forma, este diagrama é representado por nós conectados por associações de comunicação. Além disso, dentro de cada nó é possível conter componentes, que por sua vez, podem conter classes, bibliotecas ou executáveis.

A Figura 2.23 mostra um diagrama de implantação com os componentes necessários para gerar o componente executável (*script* OTcl) que corresponde a um modelo de simulação de um experimento para redes *Ad Hoc*.

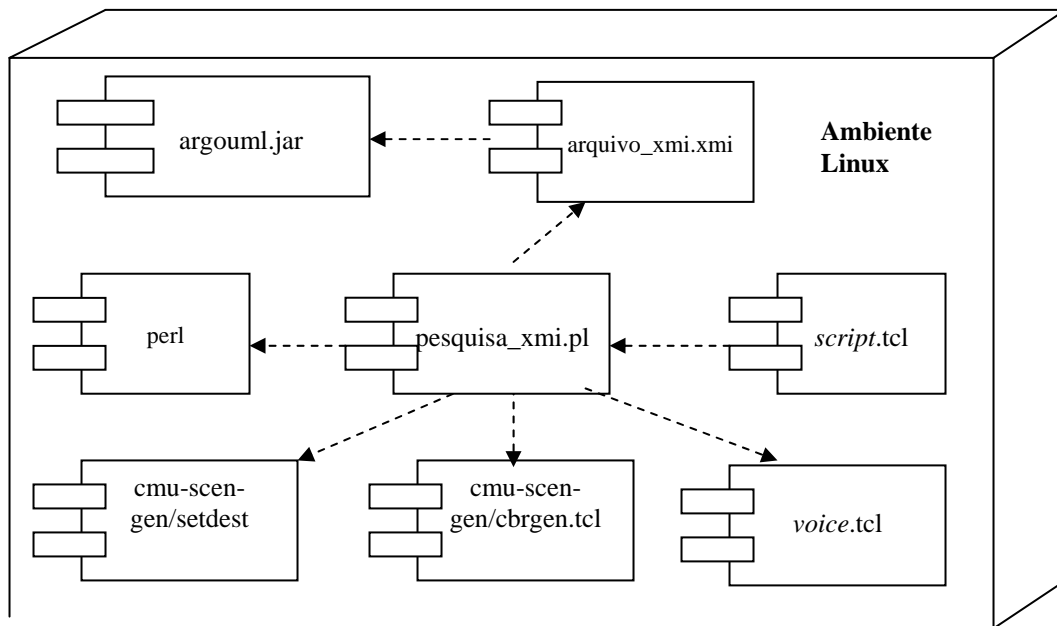



Figura 2.23: Diagrama de implantação

2.3.6 Notações genéricas

A UML oferece todos os diagramas necessários para visualizar, especificar, construir e documentar os artefatos para sistemas complexos, porém, como toda a linguagem a UML pode não abranger todos os elementos para resolver problemas. Em algumas circunstâncias, o engenheiro de software poderá ter a necessidade de adaptar ou estender a UML. A UML fornece mecanismos de extensão para adicionar novas propriedades e especificar uma nova semântica. Esses mecanismos servem para ajustar a UML às necessidades específicas para um determinado problema. Os mecanismos básicos empregados nos diagramas da UML são pacote, estereótipo e nota.

Pacote UML é uma representação gráfica usada para agrupar elementos de modelo. É uma técnica útil para manipular grandes porções de um sistema e pode ser empregado para mostrar a constituição física do sistema em camadas de arquitetura. É uma forma de compreender um sistema com muitos elementos, reunindo suas abstrações em grupos

maiores, demonstrando os limites de cada sub-sistema e como eles se inter-relacionam. A utilização de pacotes permitem controlar o acesso a seus conteúdos existentes na arquitetura do sistema. Um pacote é representado por um retângulo contendo uma aba como mostra a Figura 2.24.

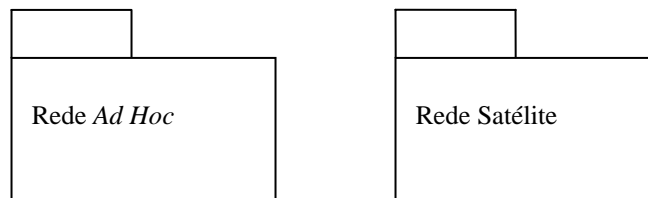



Figura 2.24: Pacotes UML

O pacote *Redes Ad hoc*, por exemplo, contém elementos empregados para geração de códigos de simulação para rede *Ad hoc* na notação *ns-2* e o pacote *Rede Satélite* contém elementos empregados para geração de códigos de simulação para redes satélite na notação *ns-2*.

Nota é um texto explicativo que pode ser acrescentado a um determinado elemento do diagrama ou ao diagrama como um todo. A nota tem a finalidade de permitir captar restrições e comentários que auxiliem a esclarecer os modelos criados. O conteúdo explicativo não tem impacto de alterar o significado do modelo ao qual está anexada. Uma nota é representada por um retângulo com uma dobra no canto superior direito.

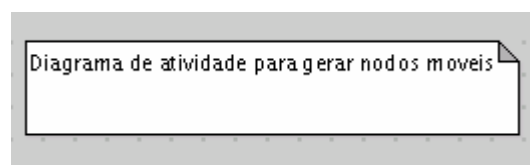



Figura 2.25 : Uma nota com um texto.

Estereótipo é um mecanismo que pode ser usado para categorizar um elemento semelhante ao outro, embora possuam características que os diferencie no diagrama. Ele é uma extensão do vocabulário da UML, que permite a criação de novos tipos de dados específicos a um determinado problema. O estereótipo auxilia a descrever mais completamente as responsabilidades de cada elemento do diagrama. Os estereótipos podem ser usados no processo de geração do código na linguagem de programação de destino. As ferramentas baseadas em UML já fornecem estereótipos prontos e permitem que outros sejam criados pelo engenheiro de software. Um estereótipo é apresentado na parte superior do componente entre os símbolos « », chamados de *guillmets* [14], acima da descrição do nome da classe. A Figura 2.26 mostra uma classe Nodo cujo estereótipo foi denominado como «Nodo» .

«Nodo» Nodo
Cor : String Nome :String Formato : String
Criar nodo ( ) Listar vizinhos ( ) ConFigurar cor ( ) ConFigurar formato ( )

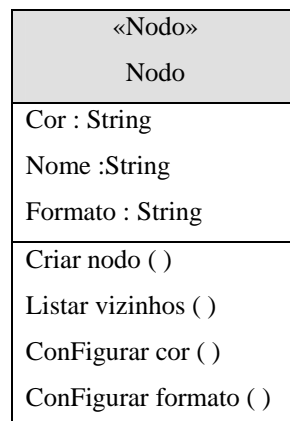



Figura 2.26: Notação do estereótipo em uma classe

Os diagramas e as notações UML podem ser modelados em várias ferramentas. Optou-se pelo ArgoUML, uma ferramenta de distribuição gratuita, mantida pela organização Tigris [15], que possibilita a modelagem visual em UML. Uma comunidade de desenvolvedores de diversas partes do mundo contribui para manutenção desta ferramenta, que oferece mecanismo de extensão, desenvolvendo novos módulos independentes da estrutura de pacotes já existentes. O ArgoUML é compatível com o padrão OMG (Object Management Group) [19] para UML e assim permite a troca de dados através de XMI entre ferramentas

UML. O XMI é um mecanismo padrão para transportar modelos UML a outras ferramentas sem perda de informação, também estabelecido pela OMG.

No Capítulo seguinte será apresentado as etapas do mapeamento de UML para *ns-2*, a análise e estudo de uma *rede Ad Hoc* para identificar elementos e os estereótipos que serão usados nos modelos de classes e atividades.

Capítulo 3

Mapeamento de UML para *ns-2*

O desenvolvimento do processo de mapeamento de UML para *ns-2* envolve estudos e análises em cenários de topologias para redes sem fio, em ferramentas de modelagem que possuam mecanismos de extensão como os estereótipos, os diagramas necessários para representar os elementos de um cenário de rede *Ad Hoc* e mecanismo de extração dos diagramas UML para padrão XMI. Além disso, a escolha de uma linguagem de programação para implementar um programa que servirá como tradutor do arquivo XMI para sintaxe do *ns-2*.

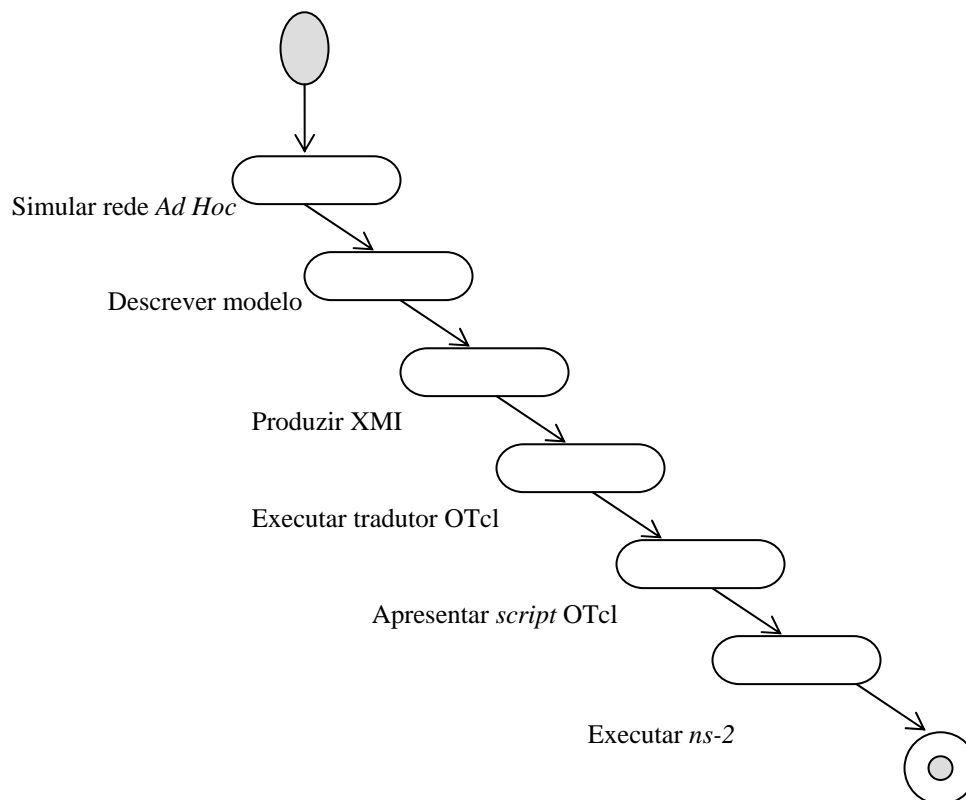



Figura 3.1: Etapas para mapeamento de UML para *ns-2*

A Figura 3.1 mostra um metamodelo[2] usando o diagrama de atividade para representar o processo de mapeamento de UML para *ns-2*. Cada estado de ação indica uma etapa desse processo. No estado de ação *Simular rede Ad Hoc*, o usuário tem um cenário de uma redes *Ad Hoc* para ser simulado. O cenário desta rede contém vários elementos que devem ser levados em consideração em uma simulação. No estado de ação *Descrever um modelo UML*, o usuário identifica os elementos envolvidos no seu cenário e descreve um modelo a partir de diagramas UML em uma ferramenta de modelagem. O usuário categoriza cada elemento do seu cenário através de um estereótipo na classe.

Após o usuário descrever o seu modelo UML, a ferramenta de modelagem faz a conversão dos diagramas UML para padrão XMI, obtendo-se o resultado do estado de ação *Produzir XMI*. O estado de ação *Executar tradutor OTcl* é um programa que pesquise no arquivo XMI, os estereótipos aplicados em cada elemento do cenário e convertê-los para notação do *ns-2*.

A execução do tradutor para OTcl produz o resultado da quinta etapa, o estado de ação *Apresentar script OTcl*. O *script* contém os elementos do cenário da rede *Ad Hoc* modelado pelo usuário na sintaxe do *ns-2*. O estado de ação *Executar ns-2* indica a etapa em que usuário executa o *script* no ambiente do *ns-2*.

3.1 Análise e estudo de uma rede *Ad Hoc*

Inicialmente, pesquisou-se vários exemplos de *scripts* de simulação disponíveis no *ns-2* e em alguns *sites* de usuários que utilizam o *ns-2*. Além disso, foram consultados alunos da graduação da UFAM que criaram seus próprios *scripts* para gerar experimentos específicos. Os alunos explicaram os *scripts* e a finalidade dos seus experimentos. A partir dessa análise, foram identificados vários elementos que podem ser simulados no *ns-2*, tais como, agentes

2 Metamodelo é uma representação dos elementos que podem existir em um modelo, seus relacionamentos e restrições de aplicação [36].

de conexão, tráfego, protocolos, movimento, área de atuação, nodos móveis, posição inicial e outros. Esses elementos serão representados por classes.

Os elementos podem usar parâmetros diferentes e ter comportamentos que são simulados aleatoriamente. Desse modo, os diagramas de classes e de atividades são mais apropriados para fazer os modelos do cenário de redes *Ad Hoc*. O diagrama de classe representa melhor os aspectos estáticos, enquanto que o de atividades representam os aspectos mais dinâmicos que são os comportamentos dos elementos que compõem o cenário. Os atributos e operações não serão utilizados no diagrama de classes no cenário, pois os estados dos atributos dos objetos das classes variam sendo melhor representá-los nos fluxos de objetos [21] em estados de ação no diagrama de atividades.

Para mostrar os relacionamentos entre as classes serão adotados agregação, composição e associação. No manual de documentação do *ns-2*, o elemento nodo para ser simulado a sua mobilidade é necesssário configurá-lo, definindo a camada MAC, interface de rede, interface de fila e o canal para citar alguns. O *ns-2* trata cada configuração como uma classe. Para descrever essas configurações no cenário modelado em UML será mapeado um classe componente, aplicando relacionamento de composição as classes referentes a configuração de um nodo móvel.

O *ns-2* trata alguns elementos com relacionamento de generalização. A classe Nodo é uma super-classe para classe Nodo Móvel. A classe Agente é uma superclasse para classe Agente TCP, Agente UDP e Agente TCPSink. A inclusão de relacionamento generalização em um cenário de experimento conduz o entendimento das hierarquias de classes e o funcionamento do *ns-2*. Na maioria dos cenários de experimentos usam elementos que aplicam conceito de superclasse. Para simplificar a construção dos elementos em modelos UML, não será adotada o uso de generalização. Uma solução encontrada é o uso do mecanismo de estereótipo.

Em [8,16,19] comentam-se que os estereótipos auxiliam no processo de geração de códigos fontes. O conceito de estereótipo será usado para ajustar os diagramas às necessidades

específicas para a geração automática do modelo de simulação em *ns-2* que se refere a quarta etapa do mapeamento de UML para *ns-2*.

Os elementos identificados podem compartilhar as mesmas características e parâmetros, mas não são necessariamente idênticos. Para ilustrar algumas situações, uma rede *Ad Hoc* usa agentes de conexão, porém, existem vários tipos de agentes, para citar exemplos TCP e UDP. O tráfego da rede *Ad Hoc* pode ser por VoIP ou CBR. Desse modo, o estereótipo será usado para categorizar mais completamente as responsabilidades de um elemento que será modelado por uma classe. Uma classe Tráfego com estereótipo «*Tráfego_VoIP*». Uma classe Agente1 com estereótipo «*Agente_TCP*» e outra classe Agente2 com estereótipo «*Agente_UDP*». Uma classe Nodo com esterótipo «*Mobile_node*».

O estereótipo também será utilizado no estado de ação para um elemento no diagrama de atividade. O estado de ação tem a função de especificar a realização do controle para o elemento. Para ilustrar algumas situações, os nodos móveis de uma rede *Ad Hoc* são criados no diagrama de atividade. Um estado de ação chamado *criar nodo* pode ser usado para criar um nodo móvel, aplicando um estereótipo «*Criar_nodo*». Uma rede *Ad Hoc* possui agentes que criam canais de comunicação entre um nodo emissor e um nodo receptor. Um estado de ação chamado *Criar agente* pode ser usado para tanto criar um agente emissor quanto a um agente receptor. Porém, é necessário aplicar estereótipos distintos para cada estado de ação, por exemplo, «*Agente_emissor*» e «*Agente_receptor*» para que o programa tradutor identifique qual é o agente emissor e o receptor.

Os estereótipos para os elementos de redes *Ad Hoc* foram capturados observando os *scripts* de simulação disponíveis no *ns-2* e exemplos de *scripts* específicos de experimentos feitos por usuários do *ns-2*.

Para melhor organização e documentação das informações coletadas foram usados modelos. Os modelos foram baseados no estudo de caso apresentado em [14]. Neste trabalho serão usados modelos para agrupar classes, estereótipos, atividades e fluxos de objetos. O modelo de classe contém o nome e uma descrição conceitual da classe que representa um elemento

usado em redes sem fio. A Tabela de estereótipos contém um nome e qual a classe que o estereótipo deverá ser aplicado. A partir do modelo e da Tabela criados para a classe, foram estentidos em outros modelos como estados de ação e fluxos de objetos para aplicar no estudo de caso deste trabalho. Os modelos e Tabelas serão explicados nas seções a seguir.

3.2 Modelo de classe

Esta Seção apresenta as principais classes de objetos identificadas durante a fase de análise nos cenários de topologias simples para redes sem fio. Em cada classe será brevemente descrito sua definição e aplicabilidade.

3.1.1 Nodo

A classe *nodo* pode ser um *desktop*, PC, roteador ou *switch* (um comutador). Esta classe pode representar uma estação de trabalho que funcione como um *host* em que os usuários executam aplicativos de programas, ou um *switch* que encaminha mensagens de um link para outro ou poderia ser um roteador que encaminha pacotes de uma rede para outra [10].

3.1.2 Mapa

A classe *mapa* representa as possíveis rotas do pacote ao longo da rede. As rotas mostram como os nodos estão conectados formando os links na rede, assim como, a simulação de falha e de recuperação de links na rede. Em redes sem fio, as rotas podem ser represesentadas pelos movimentos dos nodos móveis. A construção de uma topologia densamente povoada pode ser representada pela classe Gerador_cenário descrita no item 3.1.22.

3.1.3 Tráfego

A classe *tráfego* mostra os agentes responsáveis de criar canais de comunicação entre os nodos emissor e receptor e os serviços escolhidos à camada de aplicação que serão usados

para gerar tráfego. A construção de um tráfego com muitas conexões para uma determinada aplicação pode ser representada pela classe Gerador_tráfego descrita no item 3.1.19.

3.1.4 Aplicação

A classe A*plicação* mostra o serviço usado na camada de aplicação que gerará tráfego de informações entre os nodos. A camada de aplicação proporciona recursos para simular comportamentos de um evento tais como *enviar*, *receber*, *recuperar*, *começar* ou *parar*, e para conectá-la a um agente da camada de transporte através de funções de interface API (Applications Programming Interface) e para sinalizar se as chamadas dos procedimentos que estão no nível OTcl devem ser feitas pelos eventos *recuperar* e *receber*. Cada aplicação tem um comportamento dependendo do agente de transporte. No *ns-2* existem dois tipos básicos de aplicações que são os geradores de tráfego e "aplicações simuladas" [4]. Os geradores de tráfego são CBR (Constant Bit Rate), Pareto, Exponential e Trace, que são controladas pelo agente UPD (User Datagram Protocol) na camada de transporte. As "aplicações simuladas" são FTP (File Transport Protocol), Telnet e HTTP (HyperText Transfer Protocol). As "aplicações simuladas" referem-se a contagem de pacotes disponíveis que serão enviados pelo agente TCP na camada de transporte. As aplicações são descritas a seguir.

a) CBR é aplicado para controlar tráfego em que necessita de banda fixa (estática). Os pacotes têm tamanho constante. Geralmente são usados dados de áudio (telefonia), distribuição de áudio e vídeo, áudio e vídeo *on demand*, e jogos interativos.

b) FTP é aplicado a transferência confiável de arquivos e requer garantias de entrega de pacotes.

c) Exponential gera tráfego de acordo a uma distribuição onde os pacotes são enviados a proporção fixa durante o período de atividade, e no período de silêncio não há tráfego gerado.

d) Pareto é similar ao tráfego exponential, com a diferença de que nos períodos de atividade e de silêncio são tomados por tempos de ociosidade e rajadas pela distribuição de Pareto.

e) Telnet é um serviço utilizado para acesso remoto e exige uma transferência confiável de dados [6]. Este serviço gera pacotes de acordo com o tempo médio entre chegadas de pacotes caracterizada por uma variável chamada de intervalo. O tempo médio de chegada de pacotes pode ser escolhido a partir de uma distribuição exponential ou pela distribuição implementada chamada de *tcplib* (*tcplib-telnet.cc*).

g) HTTP é um serviço utilizado para transferência padrão da WEB [6].

3.1.5 Agente

A classe *agente* representa o agente usado na camada de transporte da rede. O agente tem a função de construir um canal de comunicação entre o nodo emissor e o nodo receptor. Existem vários protocolos que podem ser usados para a camada de transporte. Alguns protocolos não oferecem garantias para a entrega de dados entre um nodo emissor e um receptor, ou seja, não suporta controle de fluxo de congestionamento, correção de erros ou retransmissão. E outros protocolos que oferecem conexão mais confiável. Os agentes mais usados na camada de transporte são o TCP, TCPSink, LossMonitor, UDP e Null.

3.1.5.1 Agente UDP

UDP é um protocolo da camada de transporte não orientado à conexão que não oferece garantias para a entrega de dados entre um emissor e um receptor. Este protocolo não tem suporte para controle de congestionamento ou fluxo, correção de erros ou retransmissão.

3.1.5.2 Agente TCP

O protocolo TCP proporciona uma conexão confiável para a transferência de dados entre aplicações. O TCP realiza controle de fluxo e congestionamento, retransmissão, entre outras funções. O emissor TCP envia dados para um agente TCPSink e processa suas confirmações. O agente TCP pode ter as variações como Tahoe, TCP Reno e Vegas.

3.1.5.3 Agente Null

O agente Null é responsável em apenas receber pacotes sem realizar nenhum tratamento adicional.

3.1.5.4 Agente TCPSink.

A linguagem OTcl implementa conexões TCP de uma maneira, em que um emissor TCP envia pacotes de dados e um receptor TCP (conhecido como TCP Sink) envia pacotes de confirmação (ACK). O objetivo do ACK é regular a transmissão de TCP, garantindo que os pacotes sejam transmitidos somente quando o receptor devolva uma mensagem confirmando a recepção bem-sucedida.

3.1.5.5 Agente LossMonitor

O agente LossMonitor é responsável em manter estatísticas sobre os dados recebidos (número de bytes recebidos, número de pacotes perdidos e outros).

3.1.5.6 Agente RTP

O protocolo RTP proporciona serviços de entrega fim-a-fim para dados com características de tempo-real, tais como áudio interativo e vídeo. O protocolo RTP é tipicamente usado em aplicações VoIP, usando serviço de transporte de dados sem conexão.

3.1.5.7 Agente ConsBuffer

ConsBuffer é um agente implementado por alunos do laboratório wireless da UFAM para experimentos de tráfego de pacotes VoIP. Este agente tem a função de calcular o índice de referência para avaliação da qualidade da voz baseado no número de pacotes perdidos em cada segmento a partir dos parâmetros como o tempo de atraso máximo, o intervalo entre o envio de pacotes de voz que é medido pelo tamanho e o tipo do *codec* (codificação da voz).

O *codec* é um elemento importante que caracteriza a quantidade de tráfego gerado por aplicações VoIP. Existem várias opções de *codecs*, sendo que os mais utilizados são G.711, G.723.1 e G.729, que convertem a voz em formato de bits, enviando-o na forma de pacotes. G.711 é um codificador de voz de alta velocidade. G.723.1 e G.729 são codificadores de voz de baixa velocidade.

3.1.6 Variável_Aleatória (RNG-Random Number Generation)

A classe *Variável_Aleatória* pode ser usada para representar um modelo ou evento de natureza estocástica ou probabilística. O modelo ou evento é gerado a partir de variáveis aleatórias que podem ser de vários tipos de distribuição. Os tipos de distribuição mais comuns são: Uniforme, Pareto, Exponential e Constant.

3.1.7 Protocolo

A classe *protocolo* indica o protocolo de roteamento que será usado para transferir pacotes de um nodo de origem para um nodo de destino, determinando o caminho que os pacotes devem seguir. A determinação de rotas é feita por um algoritmo de roteamento. A maioria dos algoritmos em redes ad hoc são baseados em vetor de distância e podem ser agrupados de acordo com suas funcionalidades e características. Os principais algoritmos de roteamento em redes ad hoc são AODV, DSR, TORA e DSDV.

3.1.8 Nodo_Móvel

A classe *Nodo_Móvel* pode ser um *laptop*, *palmtop*, celular ou outro dispositivo móvel. Esta classe representa um dispositivo móvel.

3.1.9 Topografia

A classe *topografia* refere-se as dimensões da área de atuação dos nodos móveis.

3.1.10 Movimento

A classe *movimento* define a representação da mobilidade para um nodo móvel, indicando uma coordenada de destino e movimentando-se a uma determinada velocidade.

3.1.11 Posição

A classe *Posição* define a representação da posição inicial de um nodo em uma determinada área de atuação conhecida como topografia.

3.1.12 Componente

A classe *Componente* define a conFiguração de diferentes caracteríticas para um nodo móvel em redes sem fio. Essas características referem-se a interface da fila, camada MAC, interface de rede e quanto ao tipo de canal físico.

3.1.13 Interface Fila

A classe *Interface_Fila* determina a ocorrência de prioridade para rotear pacotes em redes sem fio, inserindo-os no início da fila, assim como o tamanho máximo da fila para suportar o roteamento de pacotes.

3.1.14 Camada MAC

À medida que cresce o número de dispositivos de comunicação e computadores portáteis, o mesmo ocorre com a demanda de conectá-los ao mundo externo. Os dispositivos de comunicação e computadores são conectados a um meio físico que pode ser sem fio ou fibra. O controle da comunicação é através de protocolo. O protocolo determina o acesso ao canal disputado entre vários usuários numa rede. Esse protocolo pertence a subcamada do nível de enlace de dados denominada subcamada MAC.

3.1.15 Interface de Rede

A classe interface da camada de rede define como os sinais são transmitidos e recebidos por antenas e tipo do modelo de rádio-propagação para determinar como o nodo móvel acessa o canal físico.

3.1.15.1 Antena

No padrão IEEE 802.11, as antenas e amplificadores são os componentes principais como meios de transmissão. As antenas convertem sinais elétricos em variações de ondas e vice-versa. Antenas “omnidirectionais” transmitem e recebem sinais de qualquer direção. Amplificadores são dispositivos que tornam sinais maiores. O aumento do sinal é medido em decibéis (dB) e podem ser classificados em: baixo ruído, alta potência ou nenhum. Os amplificadores de baixo ruído são conectados a uma antena para aumentar o sinal recebido a um nível que é reconhecido por sistemas de radiofreqüência que está conectado a ele. Os amplificadores de alta potência são usados para aumentar um sinal a uma máxima potência antes da transmissão.

3.1.15.2 Modelo de Rádio-Propagação

A propagação de freqüência de rádio é uma das técnicas para transmissão de dados através de sinais criando ondas. As informações quanto à variação de algumas propriedades físicas, como voltagem e corrente, podem ser analisadas matematicamente [22]. Uma das questões na engenharia para o enlace sem fio é codificar eficientemente o sinal para minimizar a razão de erro de bits apesar da interferência e desvanecimento. Os problemas quanto propagação de freqüência de rádio causados por interferência mais comuns são descritos a seguir:

a) Interferência de múltiplos caminhos. Este efeito causa alguns problemas como a superposição de ondas, denominado de desvanecimento por múltiplos caminhos. Desse modo, a chegada dos sinais se torna mais demorada do que as das ondas diretas.

b) Interferência de inter-símbolo. O tempo entre a chegada da primeira onda e o último eco do múltiplo caminhos é chamado de *delay spread* (atraso de dispersão).

Investigações no projeto de antenas para melhorar questões quanto espaço, freqüência, polarização, ângulo e outros itens pode ajudar a combater problemas de múltiplos caminhos.

Os modelos de rádio-propagação mais usados em simulação são o *FreeSpace* e *TwoRayRandom*. O modelo *FreeSpace* é adequado para distâncias curtas e o modelo *TwoRayRandom* é usado para distâncias longas.

3.1.16 Canal_físico

O *ns-2* simula vários tipos de redes como satélite, LAN e redes sem fio. A classe *Canal_físico* define o tipo de canal em que os pacotes são transmitidos para camada física da rede. Em redes *Ad Hoc* é usado o canal wireless.

3.1.17 Tempo

A classe *Tempo* define uma variável de tempo que pode ser usada em um evento.

3.1.18 Modelo_Descarte

A classe *Modelo_Descarte* define a conFiguração de perdas de pacotes ou erros introduzidos durante a simulação. A perda de pacotes ou erros podem ser gerados a partir de modelo estocástico ou probabilístico representados por variáveis aleatórias. Cada nodo móvel pode inserir um modelo de erro estatístico na sua entrada ou saída em um canal de rede sem fio.

3.1.19 Gerador_tráfego

Em redes sem fio, as conexões de tráfego para topologias extensas podem ser criadas a partir de um programa gerador desenvolvido pelo grupo CMU-Monarch, que encontra-se disponível dentro do *ns-2* no diretório ~*ns*/indep-utils/cmu-scen-gen/ do *ns-2* [4] .

3.1.20 Voz

A classe *Conexão* define dados de pacotes de voz que é transportado pela rede *Ad Hoc*. Os tráfegos de pacotes de voz caracterizam a conexão de uma chamada, que é dividida em segmentos. Em cada segmento é calculado o MOS (Mean Opinion Score) que mede o nível da qualidade de transmissão de voz durante o tráfego.

3.1.21 Segmento

A classe *Segmento* define um bloco parcial de dados de voz de uma conexão de uma chamada em tráfego VoIP.

3.1.22 Gerador_cenário

Em rede sem fio densamente povoada, um programa gerador de topologias cria os nodos móveis, as posições iniciais e os movimentos na topografia. O programa gerador de topologias foi desenvolvido pelo grupo CMU-Monarch e encontra-se disponível no diretório ~*ns*/indep-utils/cmu-scen-gen/setdest/ do *ns-2* [4].

3.3 Descrição dos estereótipos para o modelo de classe

O *ns-2* permite usar vários elementos tais como protocolos, agentes, links e aplicações. Para diferenciá-los, eles podem ser categorizados através de um estereótipo que auxilia no processo de geração de código na linguagem OTcl. A Tabela 3.1 apresenta uma proposta dos estereótipos que podem ser aplicados em uma classe que representa um elemento do cenário de redes *Ad Hoc* modelado em UML.

Tabela 3.1: Estereótipos das classes

Estereótipo	*Descrição*	*Aplica-se a classe*
« Nodo »	Modela um nodo.	Nodo
« Rota »	Modela a construção da rota entre os nodos e links.	Mapa
« Aplicação_CBR »	Modela tipo de serviço como CBR usado na camada de aplicação.	Aplicação
« Aplicação_FTP »	Modela tipo de serviço como FTP usado na camada de aplicação.	Aplicação
« Aplicação_Pareto »	Modela tipo de serviço como Pareto usada no camada de aplicação.	Aplicação
« Tráfego »	Modela a construção de um tráfego gerado pelos protocolos da camada de aplicação.	Tráfego
« Tráfego_VoIP »	Modela a construção de um tráfego VoIP gerado pelos protocolos da camada de aplicação.	Tráfego
« Mobile_node »	Modela um nodo móvel.	Nodo_Móvel
« Topografia »	Modela as dimensões da área de atuação dos nodos móveis.	Topografia
« Agente_NULL »	Modela agente que encarregado de receber pacotes sem realizar tratamento na camada de transporte.	Agente
« Agente_UDP »	Modela agente UDP encarregado de enviar pacotes a outros agentes na camada de transporte.	Agente
« Agente_TCP »	Modela agente TCP encarregado de enviar pacotes a outros agentes na camada de transporte.	Agente
« Agente_SINK »	Modela agente TCPSink encarregado de receber pacotes de outros agentes na camada de transporte.	Agente
« Agente_LossMonitor »	Modela agente Loss Monitor encarregado para manter estatísticas sobre os pacotes recebidos (número de bytes recebidos, número de pacotes perdidos, etc.)	Agente
« Agente_RTP»	Modela o protocolo RTP como um agente encarregado para entrega de dados com características de tempo-real como áudio interativo e vídeo.	Agente

Tabela 3.1: Estereótipos das classes

Estereótipo	*Descrição*	*Aplica-se a classe*
« Agente_Buffer»	Modela agente ConsBuffer que calcula o índice de referência para avaliação da qualidade da voz baseado no número de pacotes perdidos em cada segmento.	Agente
« RNG_Uniform »	Modela variável aleatória como distribuição uniforme	Variável_Aleatória
« RNG_Pareto »	Modela variável aleatória como distribuição Pareto	Variável_Aleatória
« RNG_Exponential »	Modela variável aleatória como distribuição Exponential	Variável_Aleatória
« RNG_Constant »	Modela variável aleatória como distribuição Constante	Variável_Aleatória
« Protocolo »	Modela um protocolo de roteamento.	Protocolo
« Movimento »	Modela um movimento de um nodo móvel	Movimento
« Posição »	Modela a posição inicial do nodo móvel dentro das dimensões da área.	Posição
« Componente »	Modela as características de conFiguração para um nodo móvel.	Componente
« Gerador_cenário »	Modela a geração automática para uma topologia extensa para criar nodos móveis, posição inicial e movimentos dos nodos.	Mapa
« Gerador_tráfego »	Modela a geração automática para uma grande quantidade de tráfegos para os nodos móveis.	Gerador_tráfego
«Tempo»	Modela uma variável de tempo	Tempo
« Descarte_pacote»	Modela a conFiguração para o descarte de pacotes ou erros durante a simulação.	Modelo_descarte
« MAC»	Modela o padrão IEEE usado para a camada MAC para redes sem fio.	Camada MAC
« Interface_rede»	Modela os componentes usados na interface de rede.	Interface_rede
« Canal »	Modela o tipo de canal usado para transmissão de pacotes para a camada física	Canal_físico
« Interface_fila»	Modela a prioridade e o tamanho da fila para rotear pacotes.	Interface_fila

Tabela 3.1: Estereótipos das classes

Estereótipo	*Descrição*	*Aplica-se a classe*
«voz»	Modela dados de voz que é transportado pela rede	Conexão
«Segmento»	Modela um bloco de dados de voz em uma conexão de uma chamada.	Segmento

3.4 Descrição das atividades para o modelo de classe

O diagrama de atividade para uma classe mostra os estados de ação que os objetos dessa classe podem assumir e as transições que eles podem fazer de estado para estado. Nesta Seção é apresentada uma proposta das possíveis atividades e os seus respectivos estados de ação para aplicar no estudo de caso de um cenário de simulação de um tráfego VoIP em redes *Ad Hoc*. Os estados de ação e seus estereótipos são descritos nas Tabelas abaixo.

3.3.1 Gerar os nodos

A atividade deve mostrar a criação de cada nodo móvel de topologia simples com poucos nodos.

Tabela 3.2: Gerar nodos

Estado de ação	*Descrição*	*Estereótipo*	*Classe*
Gerar nodo	Inicia o fluxo da atividade.	Gerar_nodo	
Criar nodo	Cria um nodo	Criar_nodo	Nodo

3.3.2 Gerar tráfego

A atividade deve mostrar a conexão com os agentes da camada de transporte, que criam canais de comunicação entre os nodos emissor e receptor, e o fluxo de um tráfego de uma determinada aplicação seja gerado e transmitido por este.

Tabela 3.3: Gerar tráfego

Estado de ação	*Descrição*	*Estereótipo*	*Classe*
Criar trafego	Inicia o fluxo da atividade.	Criar_tráfego	
Definir nodo emissor	Define o nodo para atachar a um agente emissor	Nodo_emissor	Nodo
Definir nodo receptor	Define o nodo para atachar a um agente receptor	Nodo_receptor	Nodo
Definir agente emissor	Define um agente emissor	Agente_emissor	Agente
Definir agente receptor	Define um agente receptor	Agente_receptor	Agente
Criar aplicação	Define um tipo de serviço na camada de aplicação que gera o tráfego	Criar_aplicação	Aplicação
Iniciar evento tráfego	Determina o inicio do tráfego em um determinado instante	Iniciar_evento_tráfego	Tempo Variável_aleatória
Parar evento tráfego	Determina a parada do tráfego em um determinado instante	Parar _evento_tráfego	Tempo

3.3.3 Definir protocolo

A atividade deve definir o protocolo de roteamento que caracteriza a determinação das rotas para transferir pacotes de um nodo de origem para um nodo de destino.

Tabela 3.4: Definir protocolo

Estado de ação	*Descrição*	*Estereótipo*	*Classe*
Gerar definição do protocolo	Inicia o fluxo da atividade.	Gerar_protocolo	
Definir protocolo	Define o tipo de protocolo de roteamento usado na camada de rede.	Definir_protocolo	Protocolo

3.3.4 Definir topografia

A atividade deve mostrar as dimensões da área de atuação dos nodos móveis.

Tabela 3.5: Definir topografia

Estado de ação	*Descrição*	*Estereótipo*	*Classe*
Gerar topografia	Inicia o fluxo da atividade.	Gerar_topografia	
Definir topografia	Define as dimensões da área de atuação dos nodos móveis	Definir_topografia	Topografia

3.3.5 Ajustar componentes

A atividade deve mostrar a configuração dos componentes que define várias características para nodos móveis em redes *Ad Hoc*.

Tabela 3.6: Ajustar componentes

Estado de ação	*Descrição*	*Estereótipo*	*Classe*
Ajustar componentes	Inicia o fluxo da atividade.	Ajustar_componentes	
Definir canal	Define o tipo de canal usado para transmissão do pacote para camada física	definir_canal	Canal
Definir interface da rede	Define os componentes usados para interface de rede	definir_interface_rede	Interface_rede
Definir camada mac	Define o padrão usado na camada MAC	definir_mac	MAC
Definir interface da fila	Define os componentes para caracterizar a prioridade da fila	definir_fila	Interface_fila

3.3.6 Definir rota

A atividade deve mostrar a posição inicial (uma coordenada x e y) e o movimento do nodo móvel na área de atuação. Em topologias mais extensas, a definição da posição inicial e dos movimentos para cada nodo pode ser criada pela classe Gerador_cenário.

Tabela 3.7: Definir rota

Estado de ação	*Descrição*	*Estereótipo*	*Classe*
Criar rota	Inicia o fluxo da atividade.	Criar_rota	
Criar posição	Define a posicao inicial do nodo móvel	Criar_posição	Posição
Criar movimento	Define o movimento do nodo móvel	Criar_movimento	Movimento
Definir nodo móvel	Define qual o nodo móvel está criando a posição inicial ou o movimento	Definir_nodo	Nodo

3.3.7 Definir gerador de cenário

A atividade tem a função de criar um gerador de cenário para topologias mais extensas, definindo as posições iniciais e os movimentos de cada nodo móvel dentro de uma determinada área de atuação.

Tabela 3.8: Definir gerador de cenários

Estado de ação	*Descrição*	*Estereótipo*	*Classe*
Gerar cenário	Inicia o fluxo da atividade.	gerar_cenário	
Definir cenário	Define o cenário da topologia	Define_cenário	Gerador_cenário

3.3.8 Definir gerador de tráfego

A atividade tem a função de criar um gerador de tráfegos para topologias mais extensas, definindo o tipo de conexão FTP ou CBR, a quantidade de conexões.

Tabela 3.9: Definir gerador de tráfegos

Estado de ação	*Descrição*	*Estereótipo*	*Classe*
Gerar cenário tráfego	Inicia o fluxo da atividade.	gerar_cenário_tráfego	
Definir cenário tráfego	Define o cenário do tráfego	Define_cenário_tráfego	Gerador_tráfego

3.3.9 Definir modelo de descarte

A atividade tem a função de definir a configuração do modelo de descarte de pacotes ou erros sobre redes *Ad Hoc*.

Tabela 3.10: Definir modelo de descarte de pacotes ou erros

Estado de ação	*Descrição*	*Estereótipo*	*Classe*
Gerar definição do modelo de descarte	Inicia o fluxo da atividade.	gerar_modelo_descarte	
Definir taxa de perdas de pacotes ou erros	Define a taxa de perdas de pacotes ou erros durante a simulação	definir_unidade_erro	Modelo_Descarte
Definir variável aleatória para gerar perdas de pacotes ou erros	Define um tipo de variável aleatória para gerar descartes de pacotes ou erros	definir_variável_aleatória	Variável_Aleatória
Definir agente	Define um agente da camada de transporte para receber os pacotes com erros ou descartá-los.	definir_local_descarte	Agente

3.3.10 Definir cálculo do MOS em tráfego VoIP

A atividade tem a função de definir etapas para calcular e obter o MOS para uma conexão de uma chamada em tráfego VoIP.

Tabela 3.11: Definir cálculo do MOS em tráfego VoIP

Estado de ação	*Descrição*	*Estereótipo*	*Classe*
Gerar definição do modelo para duração e intervalo para pacotes de dados de voz	Inicia o fluxo da atividade.	criar_modelo_tráfego_voip	
Definir modelo de intervalo entre segmentos	Define um modelo aleatório ou um tempo para o intervalo entre segmentos para uma conexão de uma chamada.	definir_modelo_intervalo	Tempo Variável_Aleatória
Definir modelo de duração de um segmento	Define um modelo aleatório ou um tempo para a duração do segmento em uma conexão de uma chamada.	definir_modelo_duração	Tempo Variável_Aleatória
Definir tempo atual	Define o tempo corrente para ser usado no cálculo do MOS	definir_tempo_atual	Tempo
Definir tempo parada	Define o tempo de parada para ser usado no cálculo do MOS. O tempo de parada é o tempo atual mais o tempo de duração.	definir_tempo_parada	
Gerar MOS	Define o cálculo do MOS. Este estado de ação gera a chamada do procedimento usado no modelo markoviano de N-C.Chuah [24]	gerar_mos	
Produzir resultado MOS	Define a geração do resultado do MOS do segmento	produzir_resultado_mos	
Definir segmento	Define a quantidade de cálculos para um segmento.	definir_segmento	Segmento

3.3.11 Definir modelo de geração de tráfego VoIP

A atividade tem a função de definir modelo de distribuição que representa o período de atividade e de silêncio (ON-OFF) de tráfego de pacotes de dados de voz na rede *Ad Hoc*.

Tabela 3.12: Definir modelo de geração de tráfego VoIP

Estado de ação	*Descrição*	*Estereótipo*	*Classe*
Criar modelo de distribuição	Inicia o fluxo da atividade.	criar_modelo_distribuição	
Definir período de atividade	Definir período de atividade em que está trafegando dados de voz	Período_atividade	Período
Definir período de silèncio	Definir período de silêncio em que não está trafegando dados de voz	Período_silêncio	Período

3.5 Modelo de fluxo de objetos da classe

O fluxo de objeto é usado para descrever o estado de uma instância da classe [21] . Neste trabalho, o fluxo de objeto será usado para indicar os parâmetros de um elemento do cenário. Em cada fluxo é dado um nome do objeto da classe escolhida e adicionados os valores para a instância do objeto em um estado de ação da atividade. Esta Seção apresenta as Tabelas informando quais os parâmetros que podem ser usados para cada objeto de uma classe, ou seja, no fluxo de objeto da classe escolhida. A coluna estereótipo indica o nome do estereótipo da classe do objeto, a coluna campo indica o nome do parâmetro do objeto, a coluna valor indica o conteúdo a ser passado para o parâmetro e a coluna descrição indica o significado do parâmetro do objeto.

Tabela 3.13: Fluxo de objeto para classe Nodo

Estereótipo	*Campo*	*Valor*	*Descrição*
Nodo	Cor	Red, blue, yellow, gold, green, tan	Define a cor do nodo que mostrará gráfico *nam*
	Nome	String	Define o nome do nodo que mostrará no gráfico *nam*
	Formato	Círculo, caixa	Define o formato do nodo que mostrará no gráfico *nam*

Tabela 3.14: Fluxo de objeto para classe Aplicação CBR

Estereótipo	*Campo*	*Valor*	*Descrição*
Aplicação_CBR	Tamanho_pacote	Número	Define o tamanho do pacote (em bytes) .
	Taxa	Número	Define a taxa de transferência dos pacotes (em bits/s).
	Intervalo	Número	Define o intervalo de envio entre os pacotes e não deve ser usada em conjunto com a taxa de envio.

Tabela 3.15 : Fluxo de objeto para classe Aplicação FTP

Estereótipo	*Campo*	*Valor*	*Descrição*
Aplicação_FTP	Tamanho_pacote	Número	Define o tamanho do pacote (em bytes) .
	Taxa	Número	Define a taxa de transferência dos pacotes (em bits/s).
	Intervalo	Número	Define o intervalo de envio entre os pacotes e não deve ser usada em conjunto com a taxa de envio.
	Quantidade_pacotes	Número	Define a quantidade de máxima de pacotes para entrega.

Tabela 3.16: Fluxo de objeto para classe Protocolo

Estereótipo	*Campo*	*Valor*	*Descrição*
Protocolo	Protocolo	AODV, DSR, DSDV	Define o protocolo de roteamento

Tabela 3.17: Fluxo de objeto para classe Agente UDP

Estereótipo	*Campo*	*Valor*	*Descrição*
Agente_UDP	Tamanho_pacote	Número	Define o tamanho do pacote (em bytes) . O valor padrão é 1000.
	Id_fluxo_pacote	Número	Define que o fluxo de pacotes é originado de um agente UDP. O valor padrão é 0.
	Tempo_vida	Número	Define o número máximo de nodos por onde o pacote pode passar antes de ser descartado. O valor padrão é 32.

Tabela 3.18: Fluxo de objeto para classe Agente TCP

Estereótipo	*Campo*	*Valor*	*Descrição*
Agente_TCP	Tamanho_janela	Número	Define o tamanho máximo da janela da conexão TCP. O valor padrão é 20.

Tabela 3.18: Fluxo de objeto para classe Agente TCP

Estereótipo	*Campo*	*Valor*	*Descrição*
Agente_TCP	Tamanho_pacote	Número	Define o tamanho (em bytes) que o pacot e vai possuir. O valor padrão é 1000.
	Tamanho_janela_ Congestionamento	Número	Define o tamanho máximo da janela de congestionamento (em número de pacotes) da conexão TCP. O valor padrão é 0.
	Tamanho_inicial_ Janela	Número	Define o tamanho inicial da janela de congestionamento (em número de pacotes) da conexão TCP. O valor padrão é 1.
	Tamanho_rajada	Número	Define o número máximo de pacotes que o emissor pode enviar em resposta a um sinal ACK. O valor padrão é 0.

Tabela 3.19: Fluxo de objeto para classe Agente Sink

Estereótipo	*Campo*	*Valor*	*Descrição*
Agente_TCPSink	Tamanho_janela	Número	Define o tamanho máximo da janela da conexão TCP. O valor padrão é 20.
	Tamanho_pacote	Número	Define o tamanho (em bytes) que o pacote vai possuir. O valor padrão é 1000.
	Tamanho_janela_ congestionamento	Número	Define o tamanho máximo da janela de congestionamento (em número de pacotes) da conexão TCP. O valor padrão é 0.
	Tamanho_inicial_ Janela	Número	Define o tamanho inicial da janela de congestionamento (em número de pacotes) da conexão TCP. O valor padrão é 1.
	Tamanho_rajada	Número	Define o número máximo de pacotes que o emissor pode enviar em resposta a um sinal ACK. O valor padrão é 0.

Tabela 3.20: Fluxo de objeto para classe Topografia

Estereótipo	*Campo*	*Valor*	*Descrição*
Topografia	X	Número	Define a coordenada x da área de atuação dos nodos móveis.
	Y	Número	Define a coordenada y da área de atuação dos nodos móveis.

Tabela 3.21: Fluxo de objeto para classe Movimento

Estereótipo	*Campo*	*Valor*	*Descrição*
Movimento	X	Número	Define a coordenada x do movimento.
	Y	Número	Define a coordenada y do movimento.
	Tempo	Número	Define o tempo que o nodo móvel inicia o movimento.
	Velocidade	Número	Define a velocidade que o nodo se move.

Tabela 3.22: Fluxo de objeto para classe Variável Aleatória *Exponential*

Estereótipo	*Campo*	*Valor*	*Descrição*
RNG_Exponential	Média	Número	Define o número médio para uma variável aleatória.

Tabela 3.23: Fluxo de objeto para classe Variável Aleatória *Constant*

Estereótipo	*Campo*	*Valor*	*Descrição*
RNG_Constant	Valor	Número	Define um valor para uma variável aleatória.

Tabela 3.24 : Fluxo de objeto para classe Variável Aleatória *Pareto*

Estereótipo	*Campo*	*Valor*	*Descrição*
RNG_Pareto	Média	Número	Define o número médio para uma variável aleatória.

Tabela 3.25: Fluxo de objeto para classe Variável Aleatória *Uniform*

Estereótipo	*Campo*	*Valor*	*Descrição*
RNG_Uniform	Mínimo	Número	Define o menor número para uma variável aleatória.
	Máximo	Número	Define o maior número para uma variável aleatória.

Tabela 3.26: Fluxo de objeto para classe Posição_inicial

Estereótipo	*Campo*	*Valor*	*Descrição*
Posição_inicial	X	Número	Define a coordenada x inicial do nodo na área de atuação antes de se movimentar.
	Y	Número	Define a coordenada y inicial do nodo na área de atuação antes de se movimentar.

Tabela 3.27: Fluxo de objeto para classe Interface_fila

Estereótipo	*Campo*	*Valor*	*Descrição*
Interface_fila	tamanho_fila	Número	Define o tamanho máximo da fila de prioridade.
	fila	Prioridade	Define o tipo de fila

Tabela 3.28: Fluxo de objeto para classe Interface_rede

Estereótipo	*Campo*	*Valor*	*Descrição*
Interface_rede	Modelo	freeSpace, TwoRayRandom Shadowing	Define o modelo de propagação dos sinais
	Antena	omniAntenna	Define o tipo de antena
	Tipo	wireless	Define o tipo de interface de rede

Tabela 3.29: Fluxo de objeto para classe Canal

Estereótipo	*Campo*	*Valor*	*Descrição*
Canal	tipo	wireless	Define o tipo de canal usado na transmissão de pacotes para camada física.

Tabela 3.30: Fluxo de objeto para classe Camada MAC

Estereótipo	*Campo*	*Valor*	*Descrição*
MAC	Mac	802_11	Define o padrão IEEE usado para redes *ad hoc.*

Tabela 3.31: Fluxo de objeto para classe Nodo_Móvel

Estereótipo	*Campo*	*Valor*	*Descrição*
Mobile_node	Cor	Red, blue, yellow, gold, green, tan	Define a cor do nodo que mostrará gráfico
	Nome	String	Define o nome do nodo que mostrará no gráfico *nam*
	Formato	Círculo, caixa	Define o formato do nodo que mostrará no gráfico

Tabela 3.32: Fluxo de objeto para classe Modelo Descarte

Estereótipo	*Campo*	*Valor*	*Descrição*
Modelo_Descarte	Taxa	Número	Define a taxa de perdas de pacotes ou erros.

Tabela 3.33: Fluxo de objeto para classe Tempo

Estereótipo	*Campo*	*Valor*	*Descrição*
Tempo	Tempo	Número	Define o instante de início ou término de algum evento.

Tabela 3.34: Fluxo de objeto para classe Agente ConsBuffer

Estereótipo	*Campo*	*Valor*	*Descrição*
Agente_Buffer	Tipo_*codec*	G711, G729	Define o tipo de codificador de voz
	Tamanho_quadro	Número	Define o tamanho do quadro
	Atraso	Número	Define o tempo de atraso máximo permitido

Tabela 3.35: Fluxo de objeto para classe Gerador_tráfego

Estereótipo	*Campo*	*Valor*	*Descrição*
Gerador_tráfego	Tipo_tráfego	CBR ou TCP	Define o tipo de serviço usado no tráfego
	Taxa	Número	Define o taxa de envio de pacotes
	Velocidade	Número	Define a velocidade em que os nodos se movimentam. Este campo deve ser informado quando o tipo do tráfego é TCP
	Número_conexão	Número	Define a quantidade de conexões desejado. Este campo deve ser informado quando o tipo de tráfego é CBR

Tabela 3.36: Fluxo de objeto para classe Segmento

Estereótipo	*Campo*	*Valor*	*Descrição*
Segmento	Quantidade_cálculo_mos	Numero	Define a quantidade de cálculos do MOS em uma conexão de uma chamada.

Tabela 3.37: Fluxo de objeto para classe Gerador_cenário

Estereótipo	*Campo*	*Valor*	*Descrição*
Gerador_cenário	Número_nodos	Número	Define a quantidade de nodos móveis existem na topologia da rede.
	Velocidade	Número	Define a velocidade em que nodos se movimentam na topologia da rede
	Pausa	Número	Define um tempo de pausa
	Tempo_total	Número	Define o tempo da simulação

3.6 Recursos de apoio para o processo de mapeamento de UML para *ns-2*

Esta Seção apresenta os recursos de apoio quanto a escolha da ferramenta de modelagem e a linguagem de programação usada para implementar um tradutor dos diagramas UML no padrão XMI para OTcl.

3.6.1 Ferramenta de modelagem para produzir os modelos UML

Adotou-se o ArgoUML para representar cenário de uma rede *Ad Hoc* em modelos UML na segunda etapa do processo de mapeamento, devido esta ferramenta proporcionar a geração do padrão XMI e a possibilidade de criar novos estereótipos específicos para redes *Ad Hoc* sem a necessidade de implementá-los no código fonte do ArgoUML. A inclusão de novos estereótipos é descrito na Figura A.2 do Apêndice A. A versão 0.19.8 e o manual de documentação do ArgoUML podem ser obtidos no endereço http://argouml.tigris.org . É uma ferramenta gratuita.

3.6.2 Tradutor para OTcl

XSLT, C++ e OTcl são alguns exemplos de linguagens de programação que podem converter os diagramas UML no padrão XMI para a sintaxe do *ns-2*.

Optou-se Perl [11] que é uma linguagem originalmente desenvolvida para manipulação de textos podendo ser aplicado para pesquisar os diagramas UML padronizados em XMI. As estruturas e os tipos usados em Perl são fáceis de usar, e não impõe limitações sobre os dados, por exemplo, tamanho das strings e vetores podem ser construídas de acordo com a capacidade de memória da máquina. Perl permite manipular a leitura e criação de arquivos em diretórios facilmente. O programa ao manipular o arquivo XMI e ao encontrar estereótipo «*Gerador_cenário*», faz a pesquisa no diretório correspondente a uma biblioteca que contém um programa gerador de topologias instalado no *ns-2* para produzir um *script* OTcl contendo

os nodos móveis, posição inicial de cada nodo e os seus movimentos aleatórios em se tratando de redes povoadas densamente.

O programa implementado em Perl não é parte da ferramenta de modelagem. O usuário executa o programa no comando *shell* no Linux ou no *prompt* no Windows.

Capítulo 4

Metodologia e estudo de caso

Uma rede *Ad Hoc* consiste de vários elementos tais como estações, agentes de transporte, tráfego, protocolos, camada MAC, movimento e topologia. A maioria desses elementos utilizam parâmetros que determinam estados e eventos que influenciam no resultado da simulação de um experimento no *ns-2*. A metodologia é uma maneira adequada de se observar os aspectos essenciais de um cenário de rede *Ad Hoc* utilizando-se de ferramentas de apoio de tal forma que seja possível descrever o experimento. Desse modo, a metodologia constitui-se de uma abordagem organizada para atingir um objetivo, através de etapas preestabelecidas.

Este Capítulo apresenta três seções. A Seção 4.1 propõe uma metodologia para descrever cenários básicos de simulação para produzir *script* OTcl envolvendo experimentos com tráfegos com ou sem VoIP usando serviços como CBR e FTP em redes *Ad Hoc*. Em cada item é mostrado as seqüências de como aplicar os modelos de classes, atividades e os estereótipos nos diagramas UML. O Apêndice A mostra brevemente a ferramenta ArgoUML por exemplo, a localização de alguns ícones como criar classe, atividade, fluxo de objeto e estereótipo. A Seção 4.2 é apresentado os recursos utilizados para descrever os diagramas UML e a linguagem de programação para produzir o *script* OTcl representando o cenário de uma rede *Ad Hoc* na linguagem do *ns-2*. A Seção 4.3 apresenta um estudo de caso para um cenário usando tráfego VoIP em rede *Ad Hoc* seguindo a metodologia proposta.

4.1 Metodologia para descrever um cenário de rede *Ad Hoc*

Basicamente, a construção de uma simulação de redes *Ad Hoc* no *ns-2* segue por três etapas [5]:

1. Criação da topologia, definindo os nodos.
2. Criação dos agentes da camada de transporte e conexão com nodos e de um serviço da camada de aplicação.
3. Comportamento dos eventos da simulação.

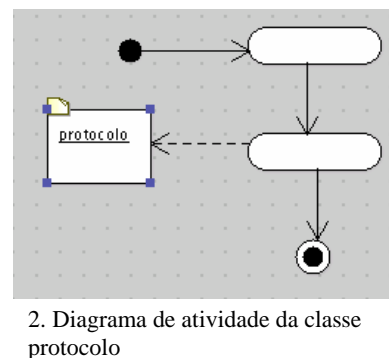
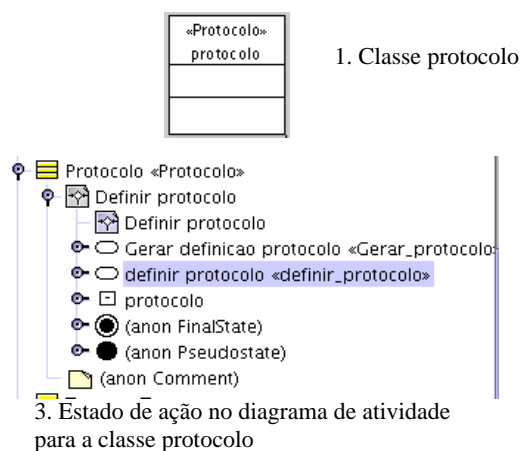
Nas subseções a seguir são descritas os aspectos importantes em cada etapa.

4.1.1 Criação da topologia da rede *Ad Hoc*

4.1.1.1 Defina o protocolo de roteamento. Os protocolos de roteamentos podem ser AODV, DSR, DSDV ou TORA que estão disponíveis no *ns-2*. Porém, o usuário pode optar por outro protocolo de roteamento, instalando-o nas bibliotecas do simulador.

1. Classe protocolo

3. Estado de ação no diagrama de atividade para a classe protocolo

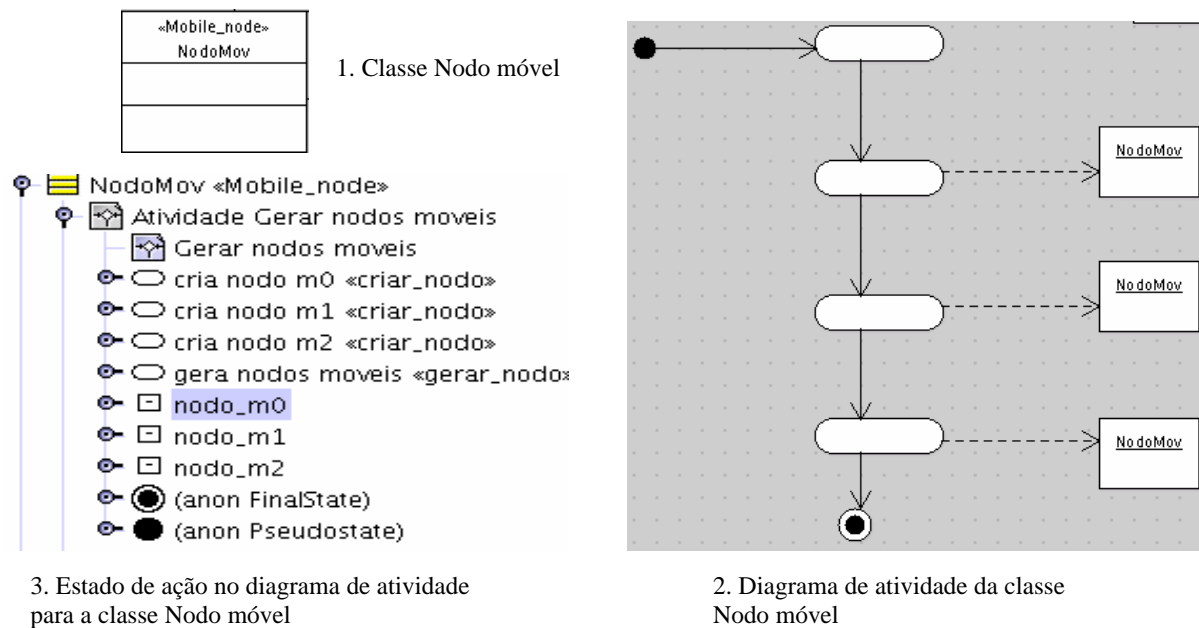
2. Diagrama de atividade da classe protocolo

4. O nome do protocolo é colocado no *tagged value* do fluxo de objeto da classe

Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar uma classe para o elemento protocolo de roteamento, denominando-a com um nome na aba *properties*. Crie o estereótipo «*Protocolo*» ou selecione-o na opção *Apply Stereotypes* como mostra na Figura A.2.

Seqüência 2: Escolha na barra de seleção de diagramas o ícone para criar um diagrama de atividades desta classe. Inicie o diagrama colocando o estado inicial com o símbolo de um círculo sólido, crie estados de ação para iniciar o fluxo da atividade e para definir o protocolo de roteamento. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar um fluxo de objeto e direcione-o usando o seta de transição ao estado de ação que estará com o estereótipo «*Definir_protocolo*». Finalize o diagrama colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.4 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

Seqüência 4: Aplique o nome do protocolo de roteamento no campo *tagged value* do fluxo de objeto da classe protocolo. A Tabela 3.16 demonstra o nome do campo e o valor para esta classe.

4.1.1.2 Defina os nodos móveis. Em cada nodo móvel especifique o nome, a cor e o formato que deseja mostrar para objeto nodo no *nam*. O *nam* é um visualizador e animador gráfico. Em topologias mais extensas, é apropriado adotar a etapa 4.1.1.7.

3. Estado de ação no diagrama de atividade para a classe Nodo móvel

2. Diagrama de atividade da classe Nodo móvel

Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar uma classe para o elemento nodo móvel, denominando-a com um nome na aba *properties*. Crie o estereótipo «*Mobile_node*» ou selecione-o na opção *Apply Stereotypes* como mostra na Figura A.2.

Seqüência 2: Escolha na barra de seleção de diagramas o ícone para criar um diagrama de atividades desta classe. Inicie o diagrama colocando o estado inicial com o símbolo de um círculo sólido, crie estados de ação para iniciar o fluxo da atividade e para definir os nodos móveis. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar um fluxo de objeto e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Criar_nodo*». Finalize o diagrama colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.2 ou selecione-o na opção *Apply Stereotypes* e denomine um nome para o estado de ação na sua aba de propriedade (*properties*).

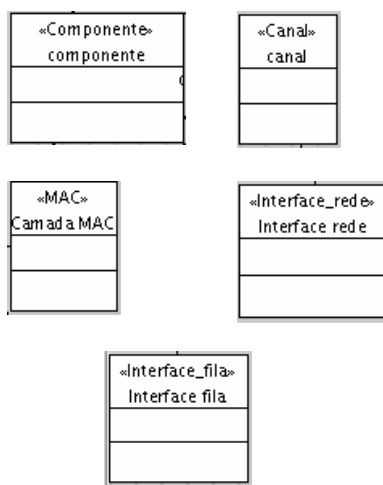
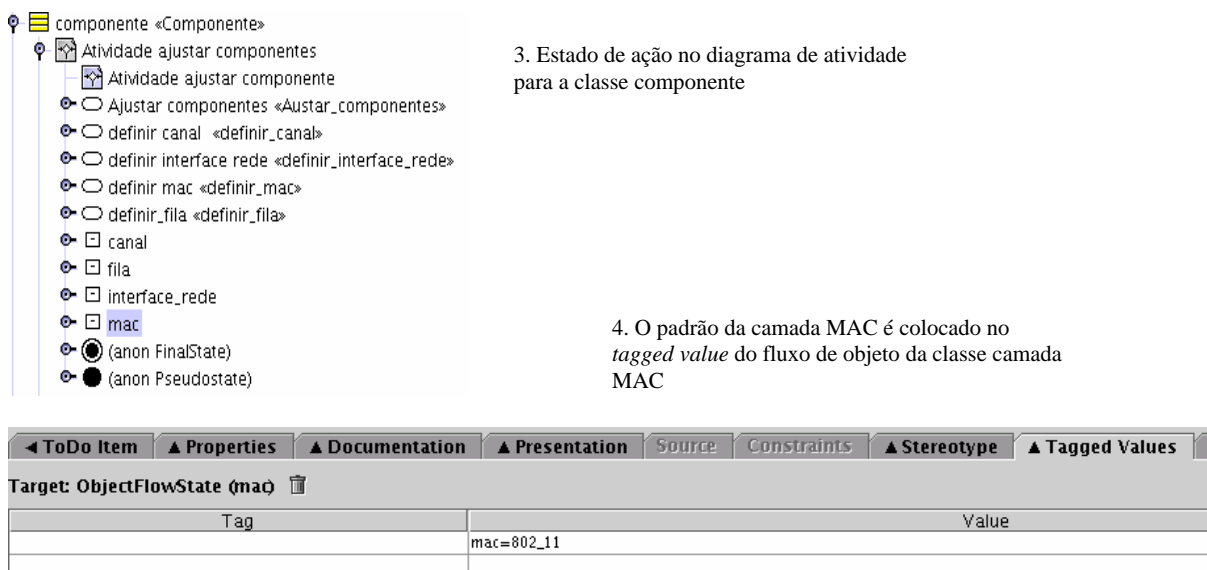
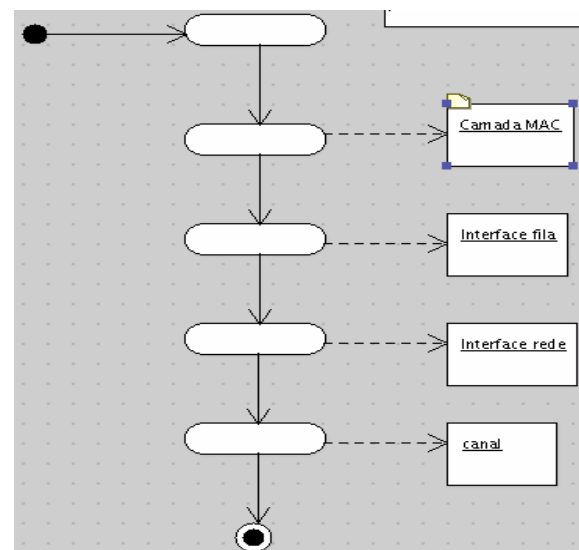
ToDo Item | Properties | Documentation | Presentation | Source | Constraints | Stereotype | Tagged Values

Target: ObjectFlowState (nodo_m0)

Tag	Value
	nome=nodo_A

4. Nome do nodo móvel nodo_m0 para mostrar no gráfico *nam*, sendo informado no *tagged value* do fluxo de objeto da classe nodo

Seqüência 4: Aplique o nome, cor e formato do nodo móvel no campo *tagged value* do fluxo de objeto da classe nodo móvel. Os campos são opcionais. A Tabela 3.31 ou Tabela 3.3 demonstra os campos e o valores para esta classe com estereótipo «*Mobile_node*» ou «*Nodo*», respectivamente.

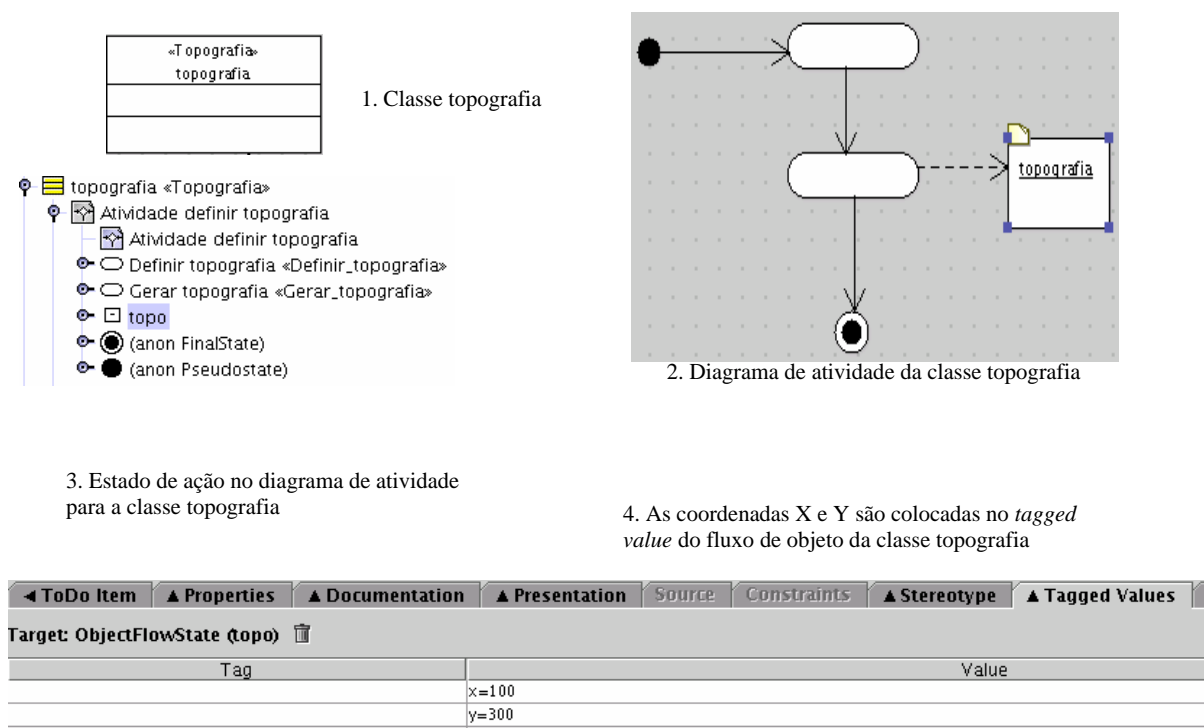
4.1.1.3 Defina a configuração do nodo móvel. Como o *ns-2* proporciona simulações para diversos tipos de redes tais como redes sem fio, satélite e LAN, esta etapa define algumas características do nodo em redes *Ad Hoc*, que referem-se a interface de rede, padrão usado para redes sem fio (IEEE 802.11) na camada MAC, prioridade adotada na fila, tipo de antena e modelo de rádio-propagação.

1. Classes : componente, Camada MAC, Canal, Interface rede e Interface fila

2. Diagrama de atividade da classe componente

Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar classes para os elementos componente, camada MAC, interface rede, interface fila e canal, denominando um nome para cada uma na aba *properties*. Crie os estereótipos «*Componente*», «*MAC*», «*Interface_rede*», «*Interface_fila*» e «*Canal*» ou selecione-os na opção *Apply Stereotypes* como mostra na Figura A.2.

Seqüência 2: Escolha na barra de seleção de diagramas o ícone para criar um diagrama de atividades para classe Componente. Inicie o diagrama colocando o estado inicial com o símbolo de um círculo sólido, crie estados de ação para iniciar o fluxo da atividade e para ajustar os componentes para caracterizar a mobilidade. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar fluxo de objeto para cada classe e direcione-o usando a seta de transição a cada estado de ação que estará com o estereótipo «*definir_mac*», «*definir_interface_rede*», «*definir_fila*» e «*definir_canal*» respectivamente. Finalize o diagrama colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.6 ou selecione-o na opção *Apply Stereotypes* e denomine um nome para o estado de ação na sua aba de propriedade (*properties*).

Seqüência 4: Aplique os valores no campo *tagged value* do fluxo de objeto das classes camada MAC, interface rede, interface fila e canal. As Tabelas 3.27, 3.28, 3.29, e 3.30 demonstram os campos e o valores para estas classes. Para exemplo, a figura desta seqüência mostra o tipo do padrão IEEE usado na camada MAC para rede *Ad Hoc* aplicando o valor 802_11.

4.1.1.4 Defina a dimensão da área dos nodos móveis no cenário da simulação, informando as coordenadas x e y.

1. Classe topografia

2. Diagrama de atividade da classe topografia

3. Estado de ação no diagrama de atividade para a classe topografia

4. As coordenadas X e Y são colocadas no *tagged value* do fluxo de objeto da classe topografia

◄ ToDo Item | ▲ Properties | ▲ Documentation | ▲ Presentation | Source | Constraints | ▲ Stereotype | ▲ Tagged Values

Target: ObjectFlowState (topo)

Tag	Value
	x=100
	y=300

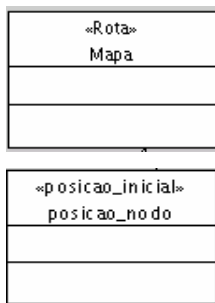
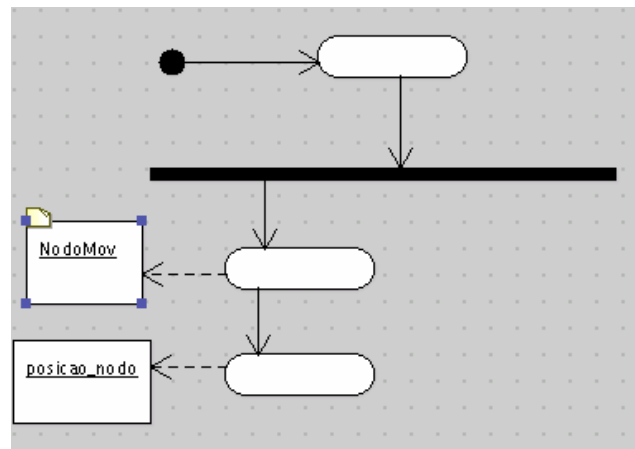
Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar uma classe para o elemento área de atuação dos nodos móveis, denominando-a com um nome na aba *properties*. Crie o estereótipo «*Topografia*» ou selecione-o na opção *Apply Stereotypes* como mostra na Figura A.2.

Seqüência 2: Escolha na barra de seleção de diagramas o ícone para criar um diagrama de atividades desta classe. Inicie o diagrama colocando o estado inicial com o símbolo de um

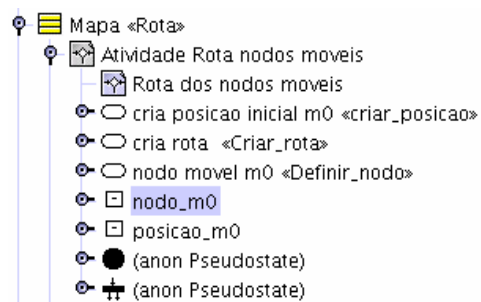
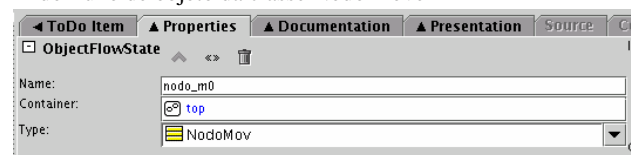
círculo sólido, crie estados de ação para iniciar o fluxo da atividade e para definir a área de atuação dos nodos. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar um fluxo de objeto e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Definir_topografia*». Finalize o diagrama colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.5 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

Seqüência 4: Aplique o nome do protocolo de roteamento no campo *tagged value* do fluxo de objeto da classe topografia. A Tabela 3.20 demonstra o nome do campo e o valor para esta classe.

4.1.1.5 Defina a posição inicial de cada nodo móvel. Um nodo móvel é designado para movimentar em uma topologia tridimensional (coordenadas x, y, z), porém, na documentação do *ns-2* cita que a coordenada z não é usada.

1. Classes: Mapa e posição_nodo

2. Diagrama de atividade da classe Mapa

Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar classes para os elementos mapa e posição inicial do nodo, denominando um nome para cada uma na aba *properties*. Crie os estereótipos «*Rota*» e «*posição_inicial*» ou selecione-os na opção *Apply Stereotypes* como mostra na Figura A.2.

Seqüência 2: Escolha na barra de seleção de diagramas o ícone para criar um diagrama de atividades para a classe Mapa. Inicie o diagrama colocando o estado inicial com o símbolo de um círculo sólido, crie um estado de ação para iniciar o fluxo da atividade e abaixo deste estado aplique o símbolo de sincronização. Abaixo da sincronização, crie estados de ação para definir a posição inicial e definir qual o nodo móvel está na posição inicial. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar fluxo de objeto para classe nodo móvel e direcione-o usando a seta de transição a cada estado de ação que estará com o estereótipo «*definir_nodo*». Escolha o ícone para criar fluxo de objeto para classe posição_nodo e direcione-o usando a seta de transição a cada estado de ação que estará com o estereótipo «*criar_posição*». Finalize esta seqüência colocando o estado final com o símbolo de um círculo sólido pequeno.

5. As coordenadas x e y são informadas no *tagged value* do fluxo de objeto da classe posição_nodo

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.7 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

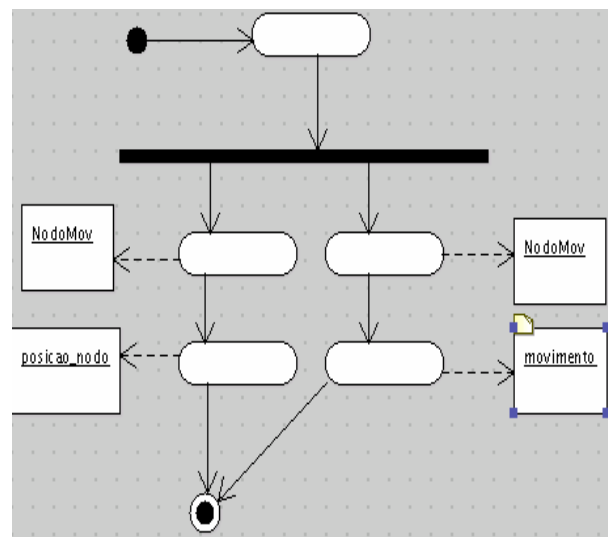
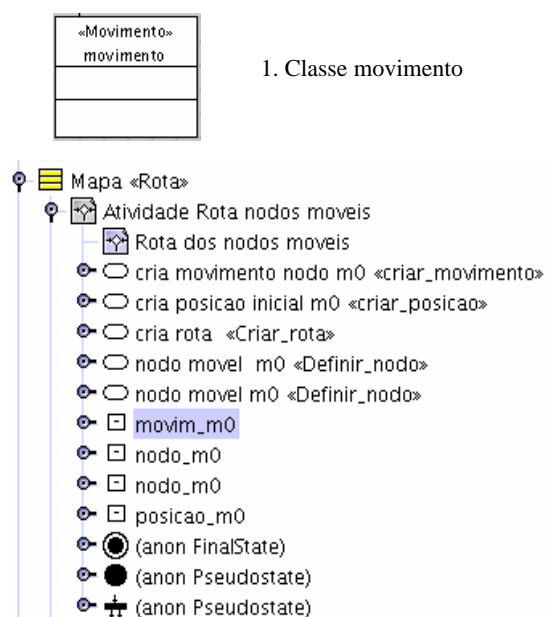
Seqüência 4: Defina o nome do nodo móvel em *properties* do fluxo de objeto da classe nodo móvel que está para uma posição inicial a ser determinada. O nome do nodo deve ser o mesmo aplicado na subseção 4.1.1.2.

Seqüência 5: Defina as coordenadas x e y que indicam a posição inicial do nodo no *tagged value* do fluxo de objeto da classe posição_nodo. A Tabela 3.26 demonstra os campos X e Y para o fluxo do objeto da classe posição_nodo.

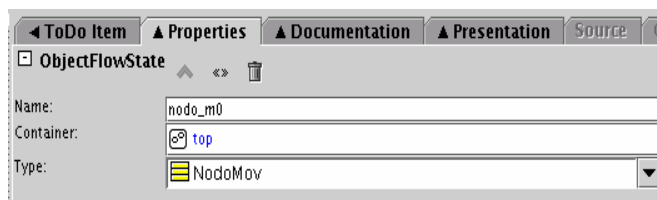
4.1.1.6 Defina a movimentação de cada nodo móvel dentro da área estabelecida na simulação, informando um coordenada (x,y) de destino, uma velocidade em que o nodo móvel está movimentando-se e um tempo que o nodo iniciou o deslocamento.

1. Classe movimento

2. Estendendo o diagrama de atividade da classe Mapa com a inclusão do elemento movimento

3. Estados de ação no diagrama de atividade para a classe Mapa

Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar a classe movimento, denominando um nome da classe na aba *properties*. Crie o estereótipo «*Movimento*» ou selecione-o na opção *Apply Stereotypes* como mostra na Figura A.2.

Seqüência 2: Utilize o mesmo diagrama de atividade da classe Mapa elaborada na subseção 4.1.1.5. Estenda o diagrama abaixo da sincronização, criando estados de ação para definir o movimento e definir qual o nodo móvel está se deslocando. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar fluxo de objeto para classe nodo móvel e direcione-o usando a seta de transição a cada estado de ação que estará com o estereótipo «*definir_nodo*». Escolha o ícone para criar fluxo de objeto para classe movimento e direcione-o usando a seta de transição a cada estado de ação que estará com o estereótipo «*criar_movimento*». Finalize a atividade colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.7 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

4. Nome do nodo móvel é informado em *properties* do fluxo de objeto da classe Nodo móvel

ToDo Item | Properties | Documentation | Presentation | Source | Constraints | Stereotype | Tagged Values

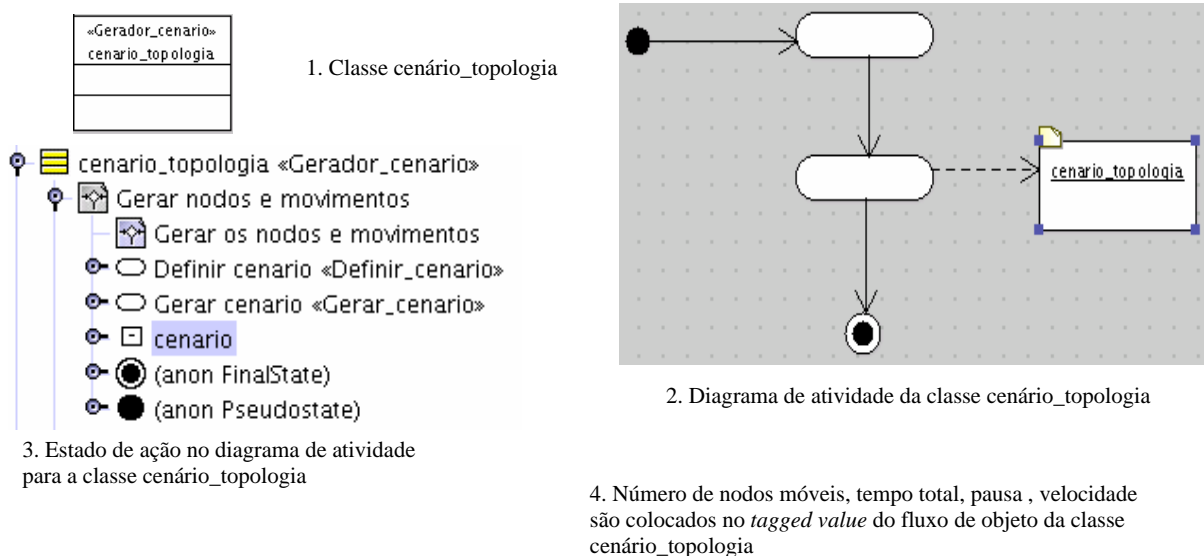
Target: ObjectFlowState (movim_m0)

Tag	Value
	x=250
	y=250
	velocidade=3
	tempo=10

5. As coordenadas X e Y, a velocidade e o tempo são colocados no *tagged value* do fluxo de objeto da classe movimento

Seqüência 4: Defina o nome do nodo móvel em *properties* do fluxo de objeto da classe nodo móvel que está para um movimento a ser determinado. O nome do nodo deve ser o mesmo aplicado na subseção 4.1.1.2.

Seqüência 5: Defina as coordenadas x e y que indicam o destino do nodo, a velocidade em que o nodo se desloca e o tempo que inicia o deslocamento, no *tagged value* do fluxo de objeto da classe movimento. A Tabela 3.21 demonstra os campos para o fluxo do objeto da classe movimento.

4.1.1.7 Defina um modelo dos movimentos dos nodos móveis. Em topologias mais extensas, um programa instalado no *ns-2* computa a criação dos nodos móveis e os seus movimentos gerando um arquivo *script* OTcl. É necessário especificar a quantidade de nodos móveis, velocidade em que os nodos móveis se movimentam, tempo total da simulação. Esta etapa pode substituir as etapas 4.1.1.2, 4.1.1.5 e 4.1.1.6.

1. Classe cenário_topologia

2. Diagrama de atividade da classe cenário_topologia

3. Estado de ação no diagrama de atividade para a classe cenário_topologia

4. Número de nodos móveis, tempo total, pausa , velocidade são colocados no *tagged value* do fluxo de objeto da classe cenário_topologia

◄ ToDo Item | ▲ Properties | ▲ Documentation | ▲ Presentation | Source | Constraints | ▲ Stereotype | ▲ Tagged Values

Target: ObjectFlowState (cenario)

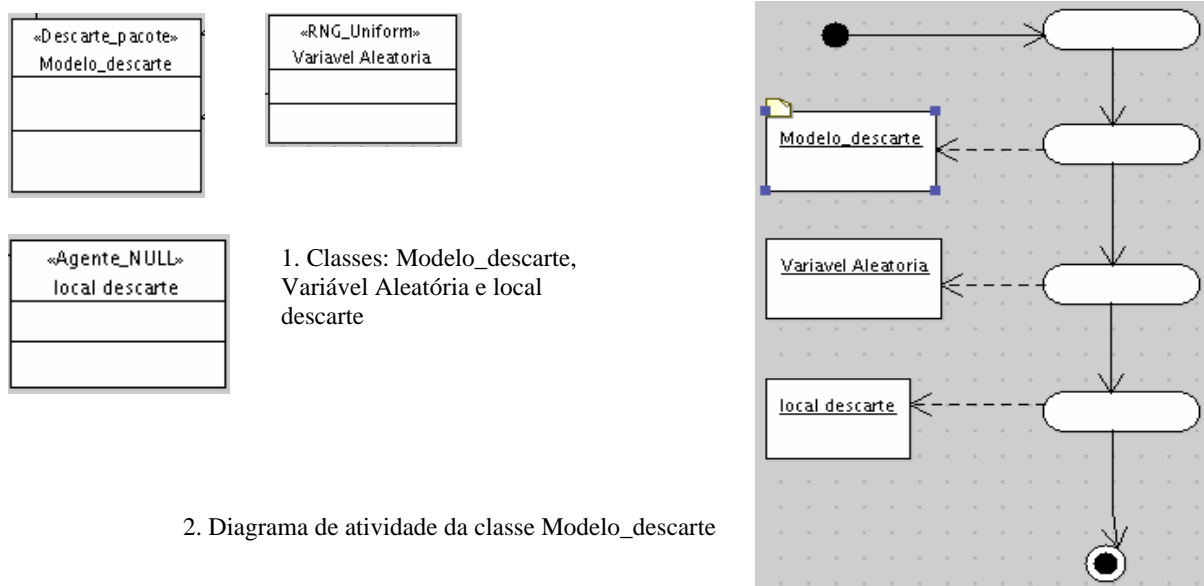
Tag	Value
	numero_nodos=30
	tempo_total=500
	pausa=10
	velocidade=30

Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar a classe cenário_topologia, denominando um nome da classe na aba *properties*. Crie o estereótipo «*Gerador_cenário*» ou selecione-o na opção *Apply Stereotypes* como mostra na Figura A.2.

Seqüência 2: Escolha na barra de seleção de diagramas o ícone para criar um diagrama de atividades desta classe. Inicie o diagrama colocando o estado inicial com o símbolo de um círculo sólido, crie estados de ação para iniciar o fluxo da atividade e para definir o cenário da topologia para uma rede densamente povoada. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar um fluxo de objeto e direcione-o usando o seta de transição ao estado de ação que estará com o estereótipo «*Definir_cenário*». Finalize o diagrama colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.8 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

Seqüência 4: Aplique os parâmetros número de nodos móveis, tempo total da simulação, pausa e velocidade dos nodos no campo *tagged value* do fluxo de objeto da classe cenário_topologia, para produzir um cenário de rede densamente povoada. A Tabela 3.37 demonstra os campos para esta classe.

4.1.1.8. Defina um modelo de descarte de pacotes com erros. O usuário pode introduzir descarte de pacotes no seu experimento, especificando um modelo de distribuição aleatória e a que taxa ocorrerão o descarte de pacotes ou pacotes com erros na entrada ou saída do nodo móvel em um canal de rede *Ad Hoc*, especificando o agente receptor desses descartes na camada de transporte. Esta etapa é opcional.

1. Classes: Modelo_descarte, Variável Aleatória e local descarte

2. Diagrama de atividade da classe Modelo_descarte

Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar classes para os elementos modelo de descarte, distribuição aleatória e agente receptor (local de descarte dos pacotes com erros), denominando um nome para cada uma na aba *properties*. Crie o estereótipo «*Descarte_pacote*» para a classe do elemento modelo de descarte. O estereótipo para a classe Variável Aleatória pode ser «RNG_Uniform», «RNG_Pareto», «*RNG_Exponential*» ou «*RNG_Constant*» como mostra na Tabela 3.1 ou selecione-os na opção *Apply Stereotypes* como mostra na Figura A.2. O estereótipo para a classe local descarte pode ser um agente receptor, como exemplo «*Agente_NULL*». A Tabela 3.1 mostra os outros estereótipos para agentes.

Seqüência 2: Escolha na barra de seleção de diagramas o ícone para criar um diagrama de atividades para classe Modelo_descarte. Inicie o diagrama colocando o estado inicial com o símbolo de um círculo sólido, crie estados de ação para iniciar o fluxo da atividade e para introduzir descarte de pacotes ou erros ocorridos na entrada ou saída do nodo móvel em um canal de rede *Ad Hoc*. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar fluxo de objeto para cada classe e direcione-o usando o seta de transição a cada estado de ação que estará com o estereótipo «*definir_unidade_erro*», «*definir_variável_aleátoria*» e «*definir_local_descarte*» respectivamente. Finalize o diagrama colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.11 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

Seqüência 4: Informe a taxa de perda ou erros de pacotes no campo *tagged value* do fluxo de objeto da classe Modelo_descarte. A Tabela 3.32 demonstra os campos para esta classe. Nesta seqüência, também aplique os parâmetros no campo *tagged value* do fluxo de objeto da classe Variável Aleatória. A Tabela 3.22, 3.23, 3.24 ou 3.25 demonstra os campos que devem ser informados no *tagged value* do fluxo de objeto da classe Variável Aleatória. Para a classe local descarte, não informe parâmetros caso o estereótipo seja «Agente_NULL», pois o agente nulo não faz tratamento na recepção de pacotes.

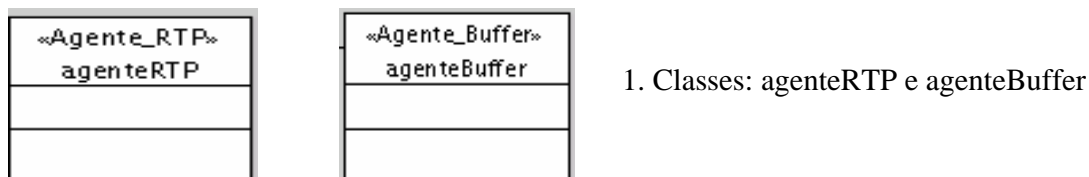
4.1.2 Definição do tráfego e dos agentes da camada de transporte da rede *Ad Hoc*.

4.1.2.1 Criação de um agente novo na camada de transporte da rede *Ad Hoc*.

O *ns-2* proporciona um conjunto de agentes prontos para usar em simulações. Porém, o usuário pode criar novos agentes, definindo uma nova classe e especificando os parâmetros

que serão usados para configurá-la durante a simulação. Esta classe deve ser implementado e compilada na linguagem C++, que resultará em um arquivo com extensão *.o*, que por sua vez deve ser adicionado na biblioteca do *ns-2* em *tcl/tools*. Os valores default dos parâmetros do novo agente devem ser adicionados na biblioteca *tcl/lib/ns-default*. O arquivo com extensão *.o* deve ser declarado no *makefile.in* do *ns-2*. O *ns-2* deve ser novamente configurado acionando o comando *.configure* e depois acionando o comando *.make* para reconhecer as alterações efetuadas nas bibliotecas do simulador.

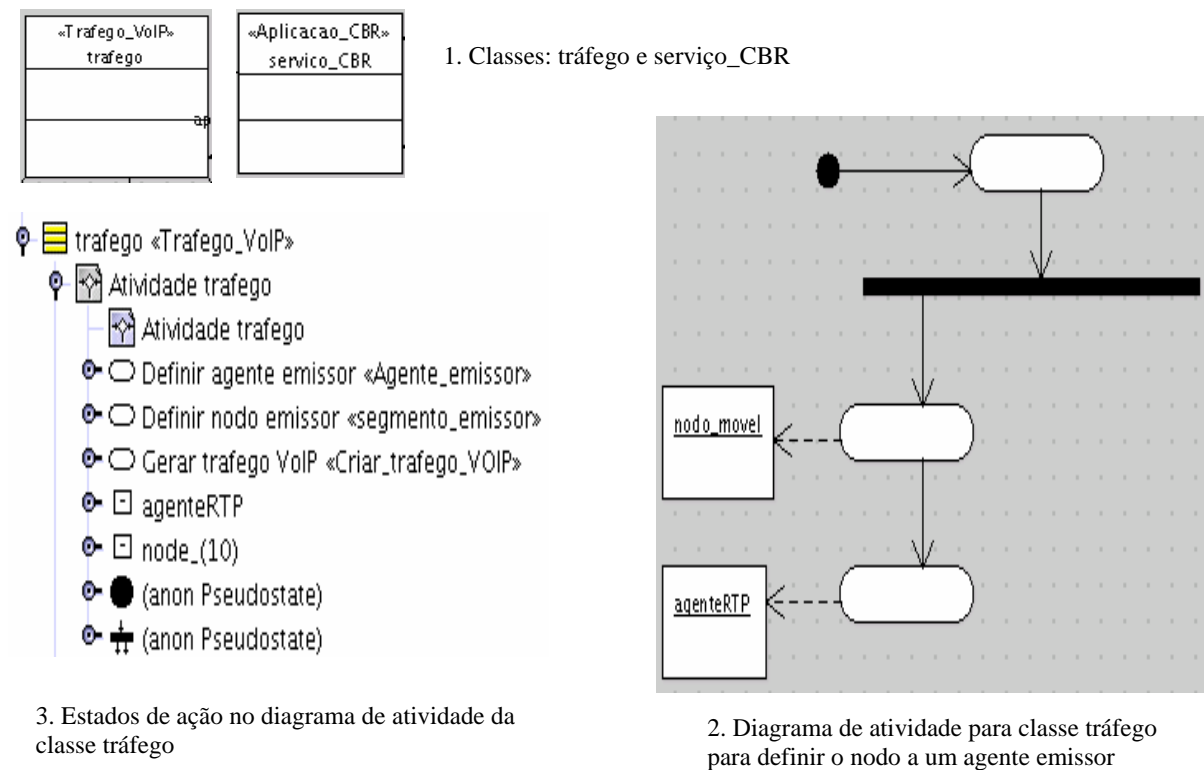
4.1.2.2 Defina os protocolos responsáveis de criar um canal de comunicação para transportar pacotes entre nodo emissor e receptor na camada de transporte.

Escolha um protocolo que desempenhará a função de um agente emissor de pacotes do nodo móvel. Escolha um protocolo com função de um agente receptor de pacotes. Os agentes LossMonitor e ConsBuffer descritos na Seção 3.1 podem ser aplicados como receptores armazenando resultados de quantos pacotes perdidos, números de *bytes* recebidos ou medir o índice de referência da qualidade de voz, conhecido como MOS (Mean Opinion Score) [25]. Os resultados podem ser gravados em um arquivo *trace* (extensão *.tr*) quando é acionado a execução do *script* OTcl da simulação. Através do arquivo *trace*, é possível extrair dados que forneçam subsídios para resolução de outros problemas em redes *Ad Hoc*. Por exemplo, se o usuário deseja obter o MOS de uma comunicação VoIP, ele pode tratar de forma *off-line* usando o arquivo trace ou tratar o MOS em tempo de execução usando um agente receptor.

1. Classes: agenteRTP e agenteBuffer

Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar as classes para um agente emissor e um receptor. Como exemplo, Agente RTP e Agente Consbuffer, respcectivamente, denominando um nome da classe na aba *properties*. Crie os estereótipos

«*Agente_RTP*» e «*Agente_Buffer*» para classe correspondente ao emissor e ao receptor, ou selecione-os na opção *Apply Stereotypes* como mostra na Figura A.2.

4.1.2.3 Defina o tipo do tráfego para a comunicação entre os agentes.

Os agentes emissor e receptor definidos na etapa 4.1.2.2 são vinculados a um nodo móvel designando-o como origem e um outro nodo móvel como o destino, respectivamente. Os agentes estabelecerão uma comunicação entre os nodos móveis da rede *Ad Hoc*. Após definir os agentes, especifique o tipo de tráfego que seja gerado e transmitido por este através de uma aplicação descrita na Seção 3.1.

1. Classes: tráfego e serviço_CBR

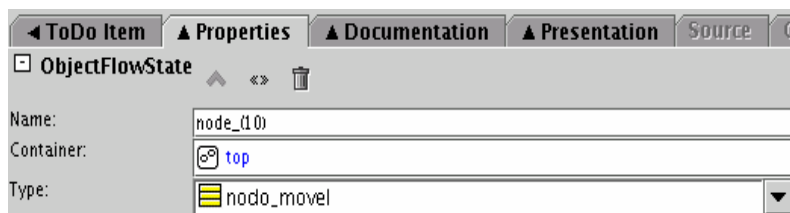
3. Estados de ação no diagrama de atividade da classe tráfego

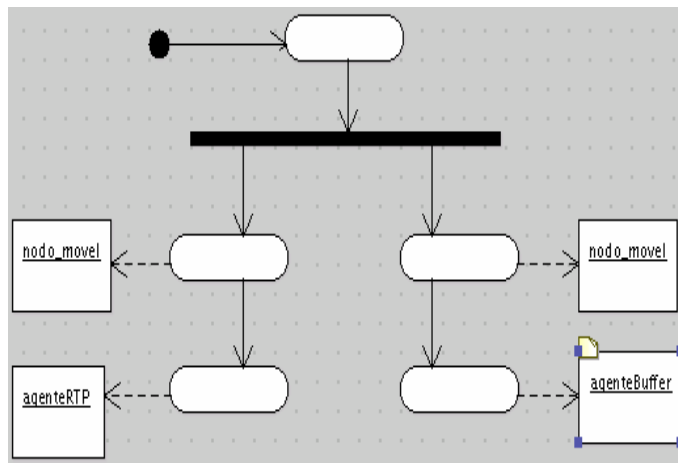
2. Diagrama de atividade para classe tráfego para definir o nodo a um agente emissor

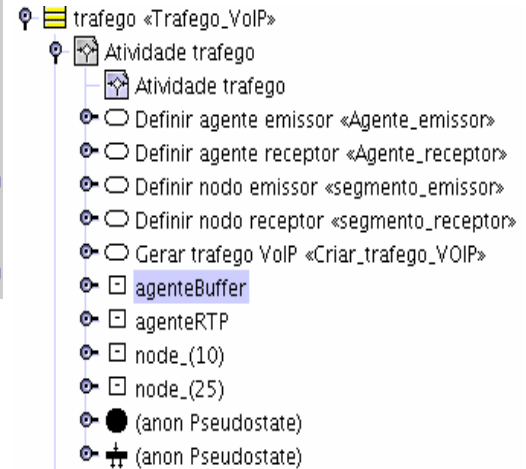
Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar as classes para um tráfego e tipo de serviço, denominando um nome da classe na aba *properties*. Crie

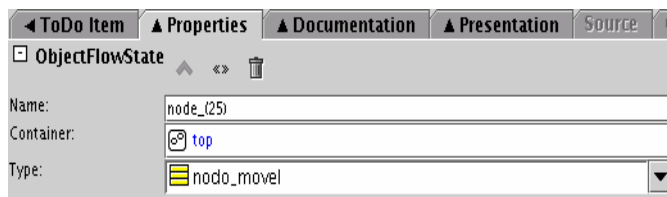
um estereótipo para a classe tráfego e um para o tipo de serviço como mostra o Tabela 3.1, ou selecione-os na opção *Apply Stereotypes* como mostra na Figura A.2.

Seqüência 2: Escolha na barra de seleção de diagramas o ícone para criar um diagrama de atividades para classe tráfego. Inicie o diagrama colocando o estado inicial com o símbolo de um círculo sólido, crie estados de ação para iniciar o fluxo da atividade e abaixo deste estado aplique o símbolo de sincronização. Abaixo da sincronização, crie estados de ação para definir o nodo emissor e o agente vinculado a este nodo. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar fluxo de objeto para classe nodo móvel e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Nodo_emissor*» ou «*segmento_emissor*» se tratando de tráfego VoIP. Escolha o ícone para criar fluxo de objeto para classe agente emissor e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Agente_emissor*».

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.3 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

4. Nome do nodo móvel como emissor é informado em *properties* do fluxo de objeto da classe Nodo móvel

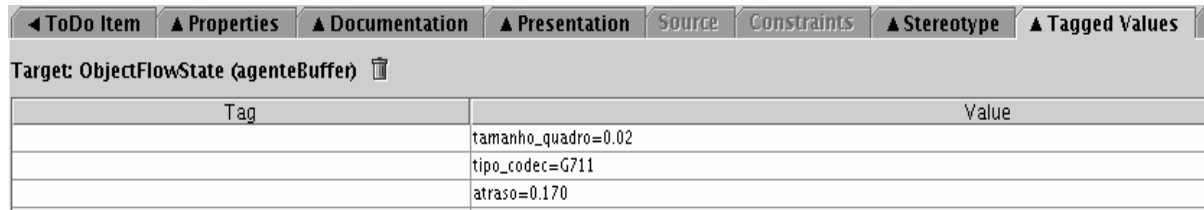
Seqüência 4: Defina o nome do nodo móvel emissor em *properties* do fluxo de objeto da classe nodo móvel. O nome do nodo deve ser o mesmo aplicado na subseção 4.1.1.2.

5. Estendendo o diagrama de atividade da classe tráfego para inclusão dos elementos nodo e agente receptor

6. Estados de ação para diagrama de atividade para a classe tráfego

Seqüência 5: Estenda o diagrama de atividade da classe tráfego abaixo da sincronização, criando estados de ação para definir o nodo receptor e o agente vinculado a este nodo. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar fluxo de objeto para classe nodo móvel e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Nodo_receptor*» ou «*segmento_receptor*» em se tratando de tráfego VoIP. Escolha o ícone para criar fluxo de objeto para classe agente emissor e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Agente_receptor*».

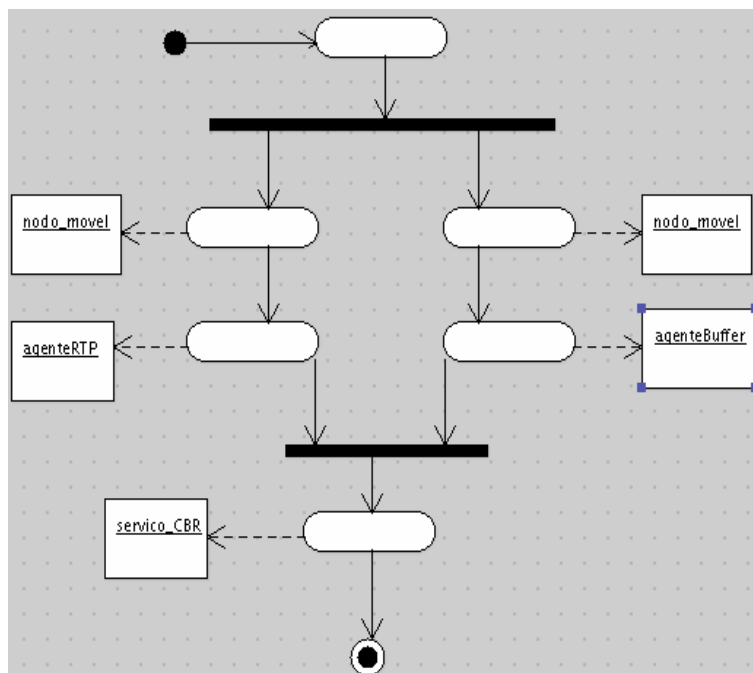
Seqüência 6: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.3 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

7. Nome do nodo móvel como receptor é informado em *properties* do fluxo de objeto da classe Nodo móvel

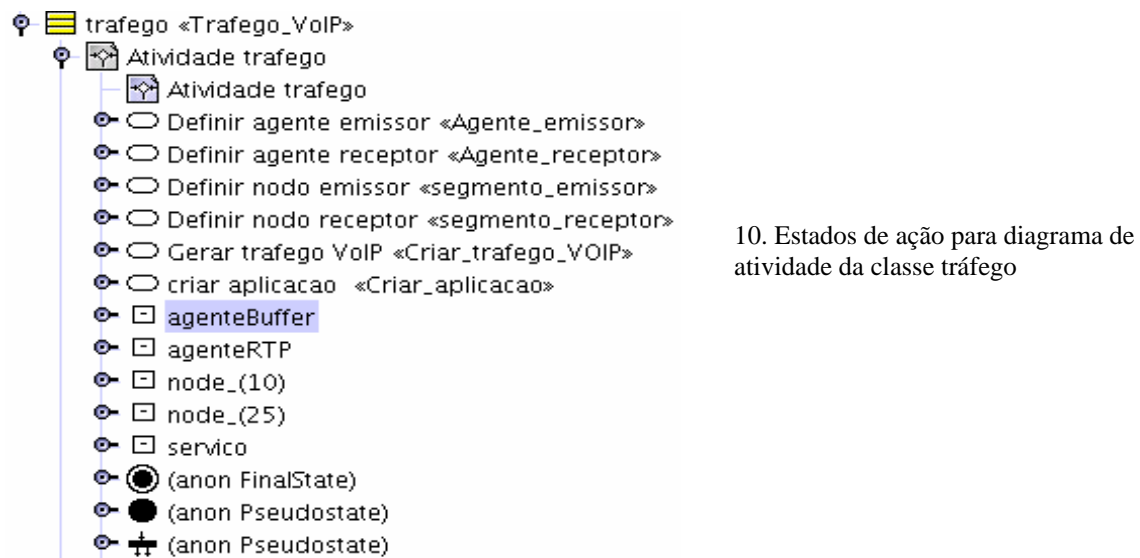
ToDo Item | Properties | Documentation | Presentation | Source | Constraints | Stereotype | Tagged Values

Target: ObjectFlowState (agenteBuffer)

Tag	Value
	tamanho_quadro=0.02
	tipo_codec=G711
	atraso=0.170

8. Tamanho do quadro, tipo do codec e tempo de atraso permitido são informados no *tagged value* do fluxo de objeto da classe agenteBuffer

Seqüência 7: Defina o nome do nodo móvel receptor em *properties* do fluxo de objeto da classe nodo móvel. O nome do nodo deve ser o mesmo aplicado na subseção 4.1.1.2.

Seqüência 8: Defina os parâmetros do agente emissor no campo *tagged value* do fluxo de objeto da classe agente receptor. A Tabela 3.19 ou 3.34 demonstra os campos que devem ser informados no *tagged value* do fluxo de objeto da classe agente escolhido no experimento.

9. Estendendo o diagrama de atividade da classe tráfego para inclusão do tipo de serviço da camada de aplicação

10. Estados de ação para diagrama de atividade da classe tráfego

Seqüência 9: Abaixo dos estados de ação dos agentes, aplique setas de transições. Crie uma sincronização e em seguida outra seta de transição. Crie um estado de ação para definir o tipo de serviço usado na camada de aplicação. Escolha o ícone para criar fluxo de objeto para classe aplicação e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Criar_aplicação*». Em se tratando de tráfego VoIP, finalize o diagrama colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 10: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.3 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

◄ ToDo Item | ▲ Properties | ▲ Documentation | ▲ Presentation | Source | Constraints | ▲ Stereotype | ▲ Tagged Values

Target: ObjectFlowState (servico)

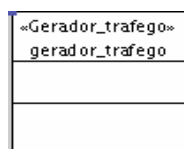
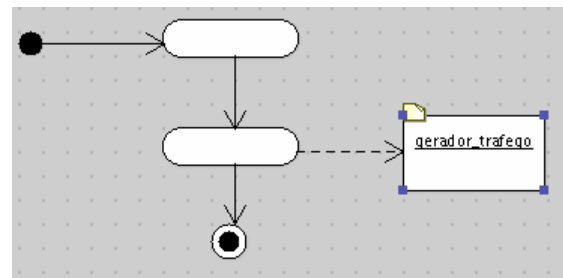
Tag	Value
	tamanho_pacote=0.5
	taxa=64

11. Tamanho do pacote e taxa de transmissão de bits são informados no *tagged value* do fluxo de objeto da classe serviço_CBR

Seqüência 11: Defina os parâmetros do tipo de serviço no campo *tagged value* do fluxo de objeto da classe serviço_CBR. A Tabela 3.14 ou 3.15 demonstra os campos que devem ser informados no *tagged value* do fluxo de objeto da classe serviço_CBR.

4.1.2.4 Defina o tráfego para topologias extensas

Em topologias mais extensas, as conexões de tráfego podem ser gerados através de um programa instalado na biblioteca do *ns-2* que computam conexões CBR ou TCP. O programa cria um cenário de tráfego com agentes emissor e receptor usando alguns nodos móveis. Segundo a documentação do *ns-2*, para gerar conexões CBR, é necessário especificar o tipo do serviço como *cbr* ou *tcp*, a quantidade de nodos móveis, velocidade, número de conexões desejado, a taxa de envio de pacotes. Para gerar conexões TCP, é necessário especificar a quantidade de nodos móveis e a velocidade que os nodos móveis se movimentam. Esta etapa pode substituir a 4.1.2.3.

Na criação do fluxo de objeto no diagrama de atividade desta etapa, a quantidade de nodos móveis pode ser capturada a partir da etapa 4.1.1.2 ou 4.1.1.7.

1. Classe gerador_tráfego

2. Diagrama de atividade da classe gerador_tráfego

Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar a classe gerador_tráfego, denominando um nome da classe na aba *properties*. Crie o estereótipo «*Gerador_tráfego*» ou selecione-o na opção *Apply Stereotypes* como mostra na Figura A.2.

Seqüência 2: Escolha na barra de seleção de diagramas o ícone para criar um diagrama de atividades desta classe. Inicie o diagrama colocando o estado inicial com o símbolo de um círculo sólido, crie estados de ação para iniciar o fluxo da atividade e para definir o gerador de tráfego para uma rede densamente povoada. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar um fluxo de objeto e direcione-o usando o seta de transição ao estado de ação que estará com o estereótipo «*Definir_tráfego_CMU*». Finalize o diagrama colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.10 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

Seqüência 4: Aplique os parâmetros do tipo de tráfego, aleatório, número de conexão e taxa no campo *tagged value* do fluxo de objeto da classe gerador_tráfego, para produzir um tráfego com muitas conexões. A Tabela 3.35 demonstra os campos para esta classe.

4.1.3 Comportamento dos eventos da simulação.

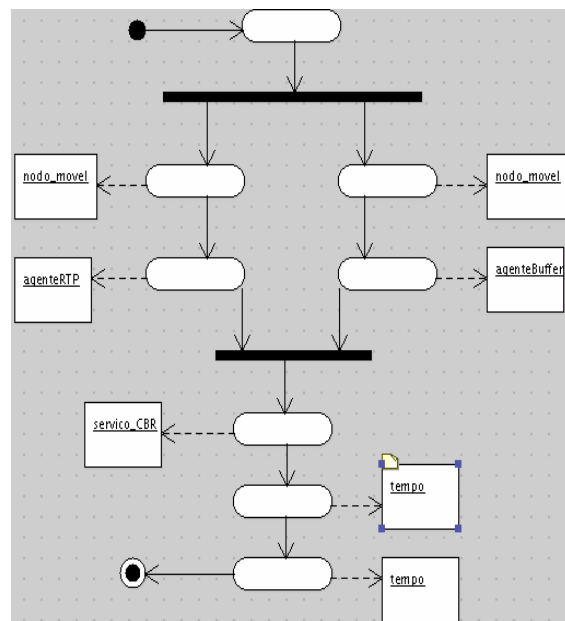
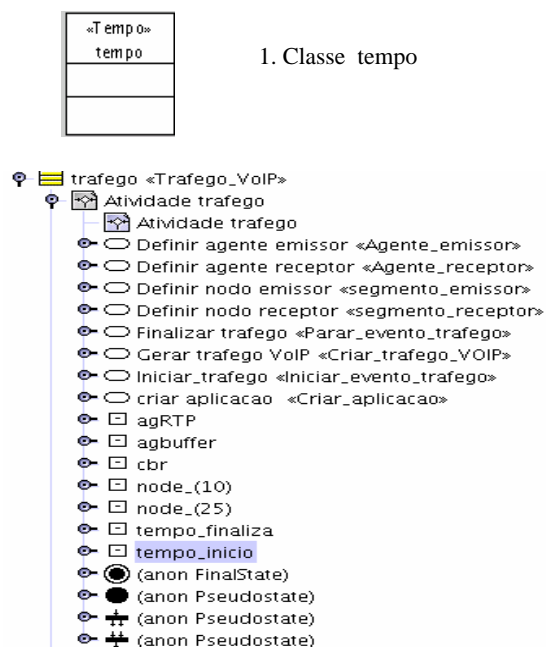
Esta etapa tem a finalidade de definir o comportamento que a simulação terá durante a sua execução. A dinâmica da simulação pode ser feita através de eventos tais como tempo total da simulação, o momento de iniciar e finalizar tráfegos, seja por um tempo fixo ou por distribuições aleatórias que modelam situações de natureza estocástica ou probabilística, períodos de intervalos e paradas. Estes foram alguns eventos que podem ser implementados e são os mais usados em um *script* OTcl, porém a implementação dos eventos da simulação depende do nível de complexidade do cenário do experimento.

4.1.3.1 Definição do instante em que inicia e termina o tráfego.

O tráfego pode ser iniciado a partir de um tempo em segundos ou pode ser disparado por um evento calculado aleatoriamente. O tráfego é finalizado indicando um tempo em segundos, por exemplo, pode ser atribuido o tempo total da simulação.

1. Classe tempo

3. Estados de ação no diagrama de atividade para a classe tráfego

2. Estendendo o diagrama de atividade da classe tráfego para inclusão do elemento tempo

Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar a classe tempo, denominando um nome da classe na aba *properties*. Crie o estereótipo «*Tempo*» ou selecione-o na opção *Apply Stereotypes* como mostra na Figura A.2.

Seqüência 2: Estenda o diagrama de atividade da classe tráfego abaixo do estado de ação cujo estereótipo é «*Criar_aplicação*», criando estados de ação para definir o tempo que inicia a simulação e o que finaliza. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar fluxo de objeto para classe tempo que indique o momento em que inicia o tráfego e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Iniciar_evento_tráfego*». Escolha o ícone para criar fluxo de objeto para classe tempo que indique o momento de parar o tráfego e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Parar_evento_tráfego*». Finalize o diagrama colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.3 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

ToDo Item | Properties | Documentation | Presentation | Source | Constraints | Stereotype | Tagged Values

Target: ObjectFlowState (tempo_inicio)

Tag	Value
	tempo=0.00001

4. O valor do tempo para iniciar o evento é informado no *tagged value* do fluxo de objeto da classe tempo

ToDo Item | Properties | Documentation | Presentation | Source | Constraints | Stereotype | Tagged Values

Target: ObjectFlowState (tempo_finaliza)

Tag	Value
	tempo=150.0

5. O valor do tempo para finalizar o evento é informado no *tagged value* do fluxo de objeto da classe tempo

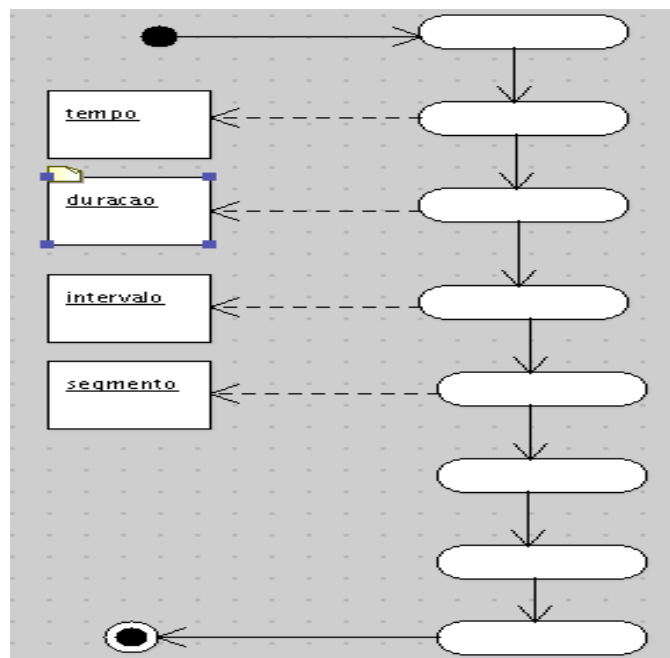
Seqüência 4: Informe o valor em segundos no campo *tagged value* do fluxo de objeto da classe tempo, para indicar o momento que inicia o tráfego. A Tabela 3.33 demonstra os campos para esta classe.

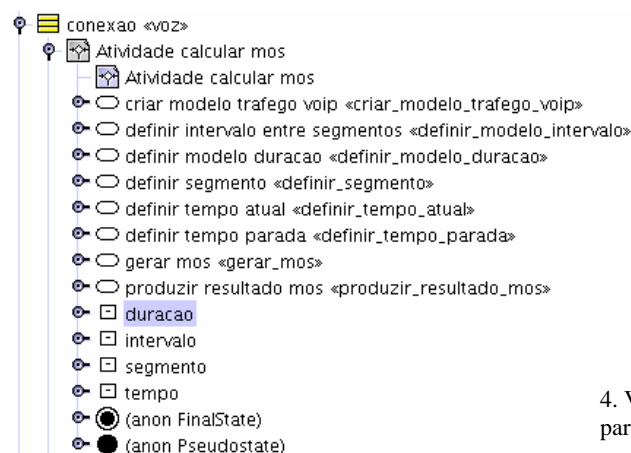
Seqüência 5: Informe o valor em segundos no campo *tagged value* do fluxo de objeto da classe tempo, para indicar o momento que finaliza o tráfego. A Tabela 3.33 demonstra os campos para esta classe.

4.1.3.2 Definição de eventos básicos para tráfego VoIP

A simulação pode ser iniciada a partir de um tempo fixo em segundos ou pelo instante atual da execução da simulação. Especifique um tempo de duração em que se está transmitindo pacotes VoIP entre dois nodos móveis. O tempo de duração pode ser calculado por um modelo de distribuição aleatória, por exemplo, distribuição uniforme. Especifique um evento que retorne o MOS calculado pelo agente receptor, caso esse agente seja o ConsBuffer. Como exemplo, o tempo para calcular o MOS é determinado pelo instante em que começou a sua execução adicionado ao tempo de duração dado por uma distribuição aleatória.

1. Classes: conexão, duração, segmento e intervalo

2. Diagrama de atividade para a classe conexão

Seqüência 1: Selecione na barra de seleção de diagramas o ícone classe para criar as classes conexão, duração, segmento e intervalo, denominando um nome para cada classe na aba *properties*. Crie um estereótipo para cada classe como mostra o Tabela 3.1, ou selecione-os na opção *Apply Stereotypes* como mostra na Figura A.2.

Seqüência 2: Escolha na barra de seleção de diagramas o ícone para criar um diagrama de atividades para classe conexão. Inicie o diagrama colocando o estado inicial com o símbolo de um círculo sólido, crie estados de ação para iniciar o fluxo da atividade, definir intervalos , definir modelo de duração, definir tempo atual, definir tempo parada, gerar MOS e produzir resultado do MOS. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar fluxo de objeto para classe tempo e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Definir_tempo_atual*» se tratando de tráfego VoIP. Escolha o ícone para criar fluxo de objeto para classe duração e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Definir_modelo_duração*». Escolha o ícone para criar fluxo de objeto para classe intervalo e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*Definir_modelo_intervalo*». Finalize o diagrama colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 3: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.11 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*). Para os estados de ação definir tempo parada, gerar MOS e produzir resultado do MOS, crie os estereótipos «*Definir_tempo_parada*», «*gerar_mos*», «*produzir_mos*», respectivamente.

Seqüência 4: Defina um tempo ou um modelo de distribuição aleatória para a duração de um segmento. Como exemplo, para usar um modelo de distribuição aleatória, defina os valores máximo e mínimo que serão aplicados em um modelo de distribuição para gerar aleatoriamente a duração de um segmento no *tagged value* do fluxo de objeto da classe duração, como mostra a figura desta seqüência. A Tabela 3.22, 3.23, 3.24 ou 3.25 demonstra os campos para o fluxo do objeto do estereótipo escolhido correspondente ao modelo de distribuição aleatória.

ToDo Item | Properties | Documentation | Presentation | Source | Constraints | Stereotype | Tagged Values

Target: ObjectFlowState (intervalo)

Tag	Value
	tempo=30.0

5. O valor do intervalo entre cada duração

ToDo Item | Properties | Documentation | Presentation | Source | Constraints | Stereotype | Tagged Values

Target: ObjectFlowState (segmento)

Tag	Value
	quantidade_calculo_mos=2

6. Quantidade de cálculos para gerar o MOS

Seqüência 5: Defina um tempo ou um modelo de distribuição aleatória para o intervalo entre cada duração de um segmento no *tagged value* do fluxo de objeto da classe intervalo. Como exemplo, a figura desta seqüência mostra que foi usado um tempo para o intervalo entre cada duração. Para usar um modelo de distribuição aleatória, a Tabela 3.21, 3.22, 3.23 ou 3.24

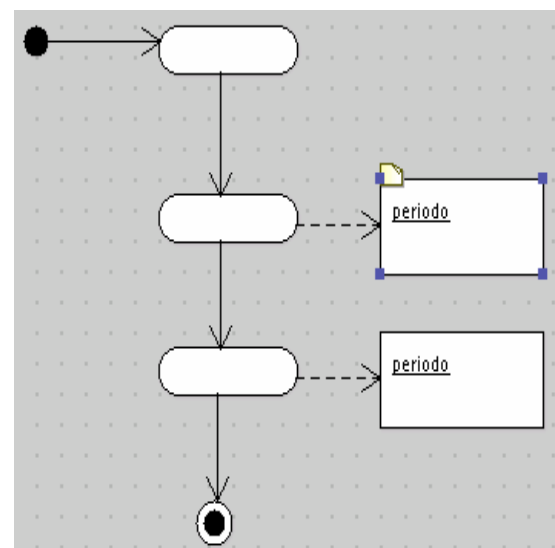
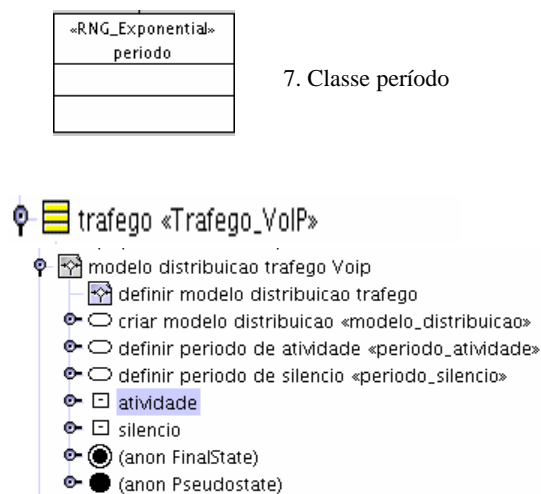
demonstra os campos para o fluxo do objeto do estereótipo escolhido correspondente ao modelo de distribuição aleatória.

Seqüência 6: Defina a quantidade de cálculos do MOS para os segmentos de dados de voz ao executar o *script* OTcl no *tagged value* do fluxo de objeto da classe segmento. A Tabela 3.36 demonstra os campos para o fluxo de objeto da classe segmento.

Em se tratando de tráfego VoIP, deve-se considerar períodos de atividade e de silêncio no canal de comunicação, ou seja, ocorrência de transmissão de voz ou não. Para representar estes períodos de atividade e de silêncio, são empregados modelos de distribuição aleatórios, o mais conhecido é o modelo de distribuição exponencial também conhecido como ON/OFF. Especifique um evento para criar um modelo de transmissão de VoIP. Este modelo representa a geração de tráfego VoIP considerando os períodos de atividade e de silêncio. Um modelo de transmissão de VoIP conhecido é o modelo markoviano de N-C.Chuah.

7. Classe período

9. Estados de ação para diagrama de atividade para representar um modelo de transmissão de VoIP na classe tráfego

8. Diagrama de atividade para representar um modelo de transmissão de VoIP na classe tráfego

Seqüência 7: Crie uma classe período no diagrama de classe e aplique o estereótipo que represente um modelo de distribuição aleatório nesta classe.

Seqüência 8: Selecione a classe tráfego e crie outro diagrama de atividade para representar modelo de transmissão do tráfego dos pacotes de dados VoIP. Crie estados de ação para iniciar o fluxo da atividade e para definir o período de atividade e o período de silêncio. Coloque a seta de transição entre os estados de ação. Escolha o ícone para criar fluxo de objeto para classe período e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*período_atividade*». Escolha o ícone para criar fluxo de objeto para classe período e direcione-o usando a seta de transição ao estado de ação que estará com o estereótipo «*período_silêncio*». Finalize o diagrama colocando o estado final com o símbolo de um círculo sólido pequeno.

Seqüência 9: Em cada estado de ação crie o estereótipo correspondente demonstrado na Tabela 3.12 ou selecione-o na opção *Apply Stereotypes* e denomine um nome na sua aba de propriedade (*properties*).

10. Média em que o tráfego está em período de atividade

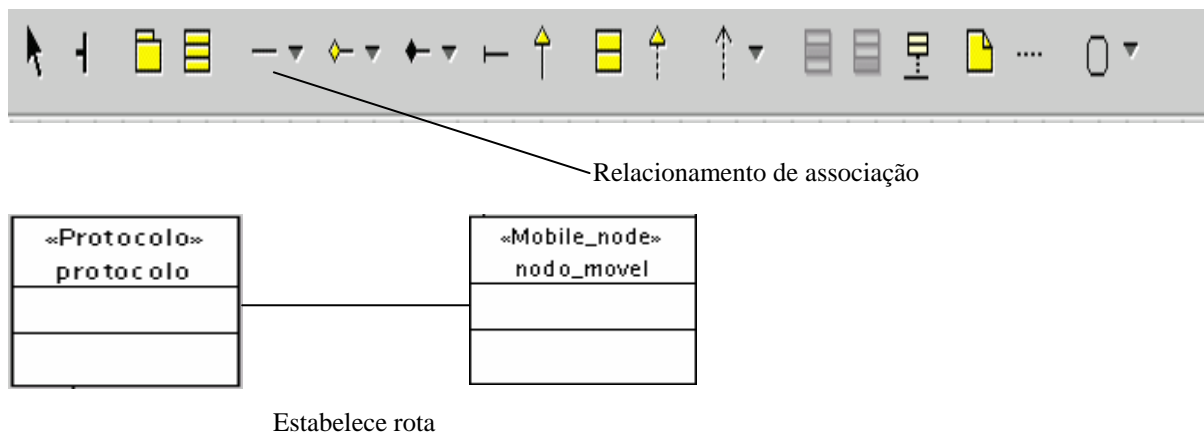
11. Média em que o tráfego está em período de silêncio

Seqüência 10: Defina a média para o modelo de distribuição aleatória escolhido no *tagged value* do fluxo de objeto da classe período, que simula o tráfego VoIP no seu período de

atividade. A Tabela 3.22, 3.23, 3.24 ou 3.25 demonstra os campos para o fluxo de objeto da classe período.

Seqüência 11: Defina a média para o modelo de distribuição aleatória escolhido no *tagged value* do fluxo de objeto da classe período, que simula o tráfego VoIP no seu período de silêncio. A Tabela 3.22, 3.23, 3.24 ou 3.25 demonstra os campos para o fluxo de objeto da classe período.

4.1.4 Aplicação dos relacionamentos entre as classes

As etapas anteriores apresentaram a criação de várias classes que são os elementos tratados em uma rede *Ad Hoc*. Utilize os conceitos de associação, agregação e composição para indicar os relacionamentos entre as classes no cenário da rede. A Figura A.2 mostra os ícones de relacionamentos que ficam disponíveis na tela de edição das classes no ArgoUML. Como exemplo, um protocolo de roteamento estabelece rota para os nodos móveis.

4.2 Estudo de Caso

4.2.1 Descrição do cenário do experimento

O cenário da simulação é uma rede *Ad Hoc* constituído de 30 estações movimentando-se em uma área de 100 × 300 metros. As estações móveis movimentam-se segundo o modelo de mobilidade *Random Way-Point* [32], transmitindo pacotes VoIP. O protocolo RTP cria um canal de comunicação para transmissão desses pacotes por taxa de bits constante (CBR) de 8 kbits/s. Os pacotes são recebidos por um agente ConsBuffer. Este agente tem a função de calcular o MOS baseado no número de pacotes perdidos através do tempo de atraso máximo permitido de 0.170 segundos e o tamanho do quadro do pacote de 0.020 segundos e o tipo de codificador de voz utilizado como G711 [33].

A duração de envio de pacotes de cada segmento é determinada por um variável aleatória, com distribuição uniforme entre os tempos de 300 a 420 segundos. O intervalo entre cada duração é determinada por tempo de 30 segundos. A duração do tráfego VoIP varia de acordo com uma variável aleatória, baseado no modelo de distribuição Exponential ON-OFF. A média da distribuição no seu período de atividade (ON) é 0.4 segundos. E no período de silêncio (OFF) é 0.6 segundos. O protocolo de roteamento é o AODV.

4.2.2 Aplicação da metodologia no ArgoUML

Esta subseção apresenta a aplicação da metodologia no ArgoUML proposta no Capítulo 4 para construir os diagramas UML para o cenário do estudo de caso.

O experimento utiliza o protocolo AODV. As seqüências do item 4.1.1.1 na etapa 4.1.1 mostram como representar o protocolo de roteamento. A topologia da rede é constituída de 30 nodos móveis. Uma classe Nodo Móvel com um estereótipo «*Mobile_node*» é criada no diagrama de classe, como cita a seqüência 1 do item 4.1.1.2 na etapa 4.1.1. Devido o experimento da rede ser densamente povoada, a posição inicial e os movimentos de cada

nodo são criados pelo programa gerador de topologias de rede. Nos parâmetros para a classe gerador, foram usados número de nodos com valor de 30, a velocidade de 30 m/s, pausa de 10 segundos e tempo total da simulação de 500 segundos. As seqüências do item 4.1.1.7 mostram a representação do gerador de topologia da rede *Ad Hoc*.

Quanto a configuração do nodo para caracterizar mobilidade, foram criados classes para camada MAC, interface da fila, interface da rede e canal. A camada MAC é definida como o padrão 802.11 por se tratar de uma rede sem fio. O tipo de fila é o *PriQueue* e o tamanho máximo do pacote na fila é no valor de 50 para uma classe representando a interface da fila. O modelo de propagação de sinais, tipo da rede e o tipo de antena são definidos como *TwoRayRandom*, *wireless* e *omniAntenna*, respectivamente. O canal é definido como *wireless*. As seqüências do item 4.1.1.3 na etapa 4.1.1 foram utilizadas para representar os componentes para configurar nodo móvel.

As estações estão movimentando-se em uma área de 100 × 300 metros. As seqüências do item 4.1.1.4 na etapa 4.1.1 foram utilizadas para representar a dimensão da área.

As estações móveis estão transmitindo pacotes de dados de voz. Na rede *Ad Hoc* está trafegando pacotes VoIP entre as estações. A Tabela 3.1 possui um estereótipo «Tráfego_VoIP» para aplicar a classe tráfego como mostra a seqüência 1 do item 4.1.2.3.

O protocolo RTP cria um canal de comunicação para transmitir esses pacotes de dados de voz, que são recebidos por um agente ConsBuffer. Desse modo, o protocolo RTP é um agente emissor e o ConsBuffer é o agente receptor. As seqüências do item 4.1.2.2 foram utilizadas para criar os agentes de comunicação para transmissão dos pacotes VoIP. O serviço usado na camada de aplicação da rede é por taxa de bits constante, conhecido como CBR. A seqüência 1 do item 4.1.2.3 mostra como criar o serviço usado na camada de aplicação da rede.

Uma estação móvel tem a função de transmitir pacotes VoIP através de seu agente pelo protocolo RTP. Outra estação móvel tem a função de recepcionar esses pacotes através do

agente ConsBuffer. As estações nodo_(10) e nodo_(25) são denominadas para serem o emissor e o receptor. Porém, outras estações podem ser escolhidas como emissor e receptor. 64 kbits/s é o valor da taxa de transmissão pelo serviço CBR. O tempo de atraso máximo, tamanho de quadro do pacote, tipo de codificador de voz são os valores para indicar no objeto da classe agente ConsBuffer. As seqüências do item 4.1.2.3 mostram como criar esses elementos nos diagrama de classe e de atividade, e aplicar os valores no fluxo de objeto correspondente a cada classe.

Após construir as seqüências do item 4.1.2.3, foi usado o mesmo diagrama de atividade da classe tráfego para definir o instante que inicia e termina a simulação, usando as sequências 4.1.3.1. Os parâmetros para o tempo que inicia é de 0.00001 segundos e o tempo que finaliza é de 500 segundos.

A finalidade do experimento é obter o nível da qualidade de voz nessa rede *Ad Hoc* que trafega pacotes de dados de voz. Uma das formas de medição é dividir o pacote com dados de voz em vários segmentos, calculando para cada um deles. Cada segmento possui um tempo de duração e entre segmentos existe um intervalo. A duração é determinada por uma função aleatória entre os tempos 300 a 430 segundos. O intervalo é determinada por um tempo de 30 segundos. As seqüências 4.1.3.2 mostram como criar esses elementos no diagrama de classe e de atividade na classe tráfego em se tratando de VoIP.

O tráfego VoIP é caracterizado por períodos de silêncio e de atividade, baseado no modelo de distribuição Exponential ON-OFF. Este modelo foi representado por uma classe Período cujo estereótipo usado foi «*RNG_Exponential*». As sequências 7 a 11 do item 4.1.3.1 mostram como criar outro diagrama de atividade na classe tráfego para representar as médias nos períodos de silêncio e de atividade.

Após criar todas as classes, as notações de associação e composição foram usadas para caracterizar os relacionamentos entre as classes no cenário do experimento. O item 4.1.4 mostra como criar um relacionamento de associação entre as classes protocolo e nodo móvel.

A Figura 4.1 mostra o diagrama de classes do cenário do estudo de caso gerado a partir das sequências dos itens comentados nesta subseção e os relacionamentos aplicados entre as classes.

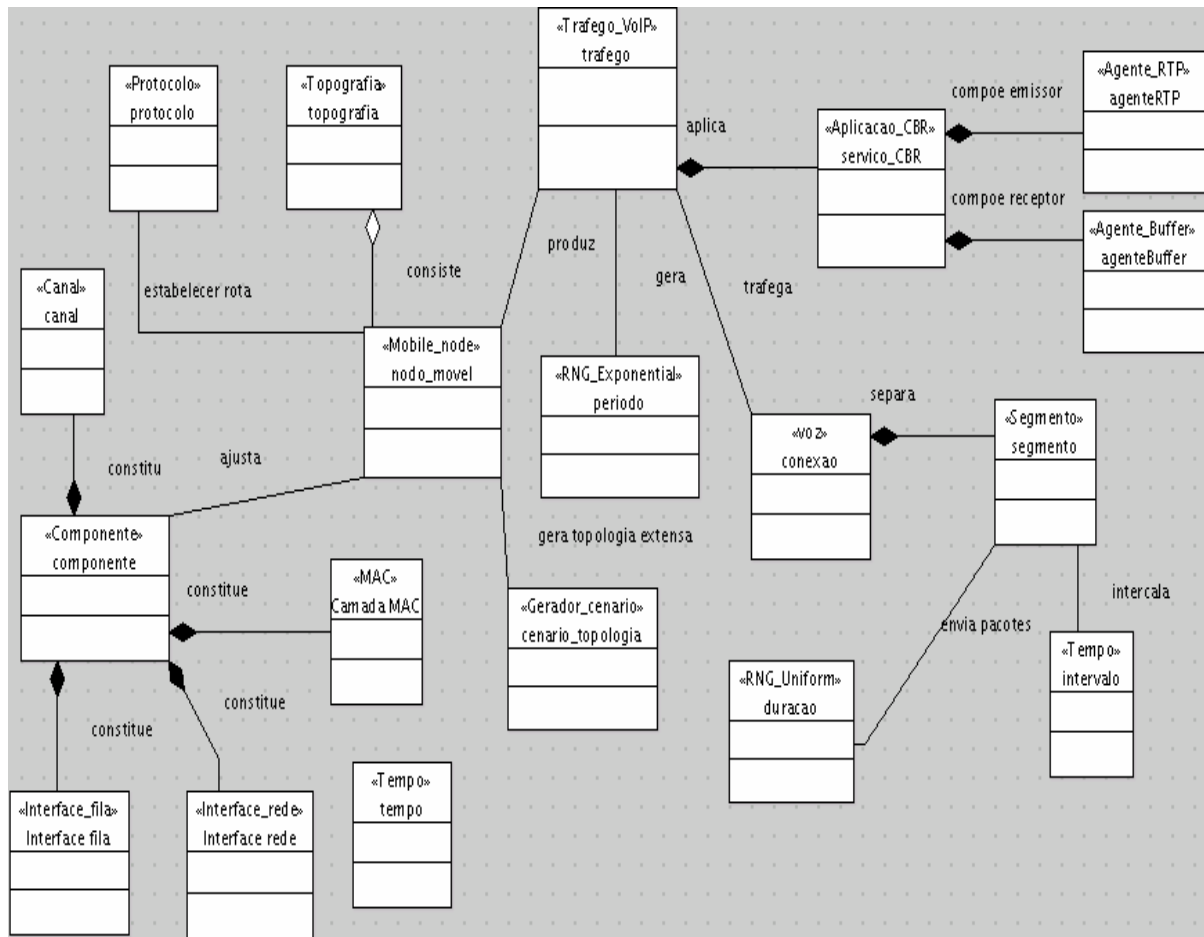



Figura 4.1 : Diagrama de classes do cenário do estudo de caso

Como em qualquer ferramenta de modelagem, o projeto deve ser salvo acionando o comando *File – Save as*. O comando *File - Export XMI* exporta o projeto para o formato XMI que conterá todos os diagramas desenhados no projeto. Um exemplo do código XMI é apresentado no Apêndice A.

4.3 *Script* OTcl do cenário do estudo de caso apoiado em UML

A Figura 4.2 mostra o *script* OTcl gerado para o cenário do estudo de caso a partir de diagramas UML.

Script OTcl

```
set ns_ [new Simulator]
set nf [open out.nam w]
set tf [open out.tr w]
$ns_ namtrace-all-wireless $nf 100 300
$ns_ trace-all $tf
set topografia [new Topography]
$topografia load_flatgrid 100 300
set expoDuration [new RandomVariable/Uniform]
$expoDuration set min_  300
$expoDuration set max_  420
set duration 0
set intervalo 30
set start 0.00001 ;# instante de inicio da primeira em uma conexão de chamada

set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen) 50
set val(nn)  30             ;#numero de nodos moveis gerador...
set god_ [create-god $val(nn)]
set chan_1 [new $val(chan)]
$ns_ node-config -adhocRouting AODV \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -ifqLen $val(ifqlen) \
                -antType $val(ant) \
              -propType $val(prop) \
              -phyType $val(netif) \
                -topoInstance $topografia \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace ON \
                -movementTrace ON \
                -channel $chan_1

for {set i 0} {$i < $val(nn)} {incr i} {
   set node_($i) [$ns_  node]
   $node_($i) random-motion 0
}
```

Figura 4.2: *Script* OTcl do cenário descrito no estudo de caso usando UML

Continuação do *Script* OTcl

```
source "Cenario_GenVoice8730.tcl"
for {set i 0} {$i < $val(nn)} {incr i} {
   $ns_ initial_node_pos  $node_($i) 30
}
set intBW 0
set audioStartTime 0.0
set audioMuInv 0.4     ;# media distribuicao exponential no periodo ON
set audioLambdaInv 0.6     ;# media distribuicao exponential no periodo OFF
set rtp0 [new Agent/RTP]
set cbr0 [new Application/Traffic/CBR]

set framelength 0.02
set audSrcBitRate 64
set buf0 [new Agent/ConsBuffer]
$buf0 set framelength 0.02
$buf0 set payloadtype G711
$buf0 set offset_ 0.170
set audioPktSize [expr round([expr $framelength * $audSrcBitRate * 1000 / 8.0])]

source  voice.tcl
set numseg 0  ;# inicializa o numero de segmentos
$ns_ at $start "makeacall 1 $numseg $node_(10) $node_(25) $buf0 $rtp0 $cbr0"

proc makeacall {numSource numseg nodeSource nodeDest buf0 rtp0 cbr0} {
   global ns_ expoDuration intervalo audioPktSize

   set numseg [expr $numseg + 1]
   set then [$ns_ now]
   set audioStartTime [expr $then]
   set duration [$expoDuration  value]
   set audioStopTime [expr $audioStartTime + $duration]

   create_audio_model $numSource $audioStartTime $audioStopTime $nodeSource
$nodeDest $audioPktSize 1 $buf0 $rtp0 $cbr0

   $buf0 start
   $ns_ at [expr $audioStopTime ] "stopcall $numSource $numseg $intervalo
$nodeSource $nodeDest $buf0 $rtp0 $cbr0"

}

proc stopcall {numSource numseg intervalo nodeSource nodeDest buf0 rtp0 cbr0} {
    global ns_
    $buf0 stop
    if { $numSource == 1 } {
    set numbuf0 [$buf0 set mos_]
    puts "resultado =>  $numbuf0"
    if { $numseg == 2 } {
          $ns_ "halt"
    }
    }
    $ns_ at [expr [$ns_ now] + $intervalo] "makeacall $numSource $numseg
$nodeSource $nodeDest $buf0 $rtp0 $cbr0"
}
```

Figura 4.2: *Script* OTcl do cenário descrito no estudo de caso usando UML

Continuação do *Script* OTcl

```
set val(stop)  500              ;#finaliza simulacao...
$ns_ at $val(stop)  "$ns_ nam-end-wireless $val(stop) "
$ns_ at $val(stop)  "stop"
$ns_ at $val(stop)  ";$ns_ halt "
proc stop { } {
    global ns_ nf tf
    $ns_ flush-trace
    close $nf
    close $tf
}
$ns_ run
```

Figura 4.2: *Script* OTcl do cenário descrito no estudo de caso usando UML

Além do *script* mostrado na Figura 4.2, o programa descrito no apêndice C gerou um *script* chamado de *Cenario_GenVoice8730.tcl* contendo todas as posições iniciais e os movimentos de cada nodo móvel, a partir da classe Gerador_Cenário. O *voice.tcl* é um *script* OTcl para geração de voz adaptado pelos alunos de graduação da UFAM para simular o tráfego de voz no modelo de distribuição exponential em períodos ON-OFF, baseado no modelo markoviano produzido por N-C.Chuah.

4.4 Avaliação e comparação dos *scripts* OTcl

Um *script* OTcl utilizando os mesmos parâmetros do estudo de caso foi escrito sem aplicar a UML, que é mostrado na Figura 4.3. A etapa do comportamento dos eventos foram também construídos de forma básica neste *script*.

Script OTcl

```
ns-random 0

# Network Systems Simulation
set ns_ [new Simulator]
```

Figura 4.3: *Script* OTcl escrito sem usar UML

Continuação do *Script* OTcl

```
set tracefd [open "/dev/null" w]

$ns_ trace-all $tracefd

set expoDuration [new RandomVariable/Uniform]
$expoDuration set min_  300
$expoDuration set max_  420
set duration 0

set acum 0 ;# variavel acumuladora para calculo do mos medio

set interval 30 ;# intervalo fixo entre chamadas consecutivas em uma comunicacao
#set calls 0
set start 0.00001 ;# instante de inicio da primeira chamada em uma comunicacao


# Link definitions
set val(chan) Channel/WirelessChannel ; # channel type
set val(prop) Propagation/TwoRayGround ; # radio-propagation model
set val(ant) Antenna/OmniAntenna ; # Antenna
set val(ll) LL ; # Link layer type
set val(ifq) Queue/DropTail/PriQueue ; # Interface queue type
set val(ifqlen) 50 ; # max packet in ifq
set val(netif) Phy/WirelessPhy ; # network interface type
set val(mac) Mac/802_11 ; # MAC type
set val(rp) AODV ; #AODV ; #DSDV ; # ad-hoc routing protocol
set val(nn) 30 ;# number of mobile nodes
set val(numsources) 1
set val(x) 100
set val(y) 300
set val(cenario) "Cenario_GenVoice8230.tcl"

# Topography definitions
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

set god_ [create-god $val(nn)]
set chan_1_ [new $val(chan)]


proc UniformErr {} {
    set uni [new ErrorModel/Uniform 0.1 pkt]

$m_transmx $m_trunit $m_sttype $m_nstates $m_nstart]
    return $uni

}

# configure nodes
$ns_ node-config -adhocRouting $val(rp)\
        -llType $val(ll)\
        -macType $val(mac)\
        -ifqType $val(ifq)\
        -ifqLen $val(ifqlen)\
```

Figura 4.3: *Script* OTcl escrito sem usar UML

Continuação do *Script* OTcl

```
        -antType $val(ant)\
        -propType $val(prop)\
        -phyType $val(netif)\
        -topoInstance $topo\
        -channel $chan_1_\
        -lossTrace OFF\
        -macTrace OFF\
        -agentTrace OFF\
        -routerTrace OFF\
        -movementTrace OFF

#Cria os nodos
for {set i 0} {$i < $val(nn)} {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) random-motion 0 ; #disable random motion

}

source $val(cenario)
for {set i 0} {$i < $val(nn)} {incr i} {
        #$node_($i) namattach $namtrace
        $ns_ initial_node_pos $node_($i) 20
}
# Voice traffic - Modelo ON/OFF

# }}}
# {{{ Source Constants
# Audio:
set audioMuInv 0.4  ;# em segundos
set audioLambdaInv 0.6 ;# em segundos
set intBW 0
set audSrcBitrate 8  ;# in kbps


set Fs 8000
set sourceBitPerSample 8

set framelength 0.020 ;# length (in time) that audio frames cover
set audioPktSize [expr round([expr $framelength * $audSrcBitrate * 1000 / 8.0])]

source "voice.tcl"

set udp [new Agent/RTP]
set cbr [new Application/Traffic/CBR]
set lossMonitor [new Agent/ConsBuffer]
$lossMonitor set offset_ 0.170
$lossMonitor framelength 0.020
$lossMonitor payloadtype G711
set numcalls 0  ;# inicializa o numero de chamadas para fonte de voz no nodo i
$ns_ at $start "makeacall 1 $numcalls $node_(10) $node_(25) $lossMonitor $udp
$cbr"

# Calls generation
proc makeacall {numSource numcalls nodeSource nodeDest lossMonitor udp cbr} {
```

Figura 4.3: *Script* OTcl escrito sem usar UML

Continuação do *Script* OTcl

```
audioStartTime audioStopTime calls framelength lossMonitor cbr udp0
audioPktSize framelength
    global ns_ expoDuration interval audioPktSize

    set then [$ns_ now]
    set duration [$expoDuration value]   ;# duracao da chamada

    set audioStartTime [expr $then]     ;# instante de inicio da chamada
    set audioStopTime [expr $audioStartTime + $duration]     ;#instante de
termino da chamada

    set numcalls [expr $numcalls + 1]
    puts "Creating call $numcalls ... source $numSource"
    #puts "duration --> $duration"

    create_audio_model $numSource $audioStartTime $audioStopTime $nodeSource
$nodeDest $audioPktSize 1 $lossMonitor $udp $cbr
    $lossMonitor start

    $ns_ at $audioStopTime "stopcall $numSource $numcalls $interval $nodeSource
$nodeDest $lossMonitor $udp $cbr"
}
proc stopcall {numSource numcalls interval nodeSource nodeDest lossMonitor udp
cbr} {
    global ns_ acum ak;#lossMonitor calls

    $lossMonitor stop
    #puts "Entrei no stopcall... source $numSource"
    if { $numSource == 1 } {
       set mos_ [$lossMonitor set mos_]
       set acum [expr $acum + $mos_]
       puts "Call $numcalls ... MOS --> $mos_\n"
       if { $numcalls == 2 } {
          set media_mos [expr $acum/$numcalls]
          puts "MOS medio --> $media_mos"
          $ns_ "halt"
       }
    }

    #$ak AkObservation $mos_
    $ns_ at [expr [$ns_ now] + $interval] "makeacall $numSource $numcalls
$nodeSource $nodeDest $lossMonitor $udp $cbr"
}

proc stop {} {
        global ns_ tracefd namtrace
        $ns_ flush-trace
        close $tracefd
        close $namtrace
}

puts "Starting simulation..."
$ns_ run
```

Figura 4.3: *Script* OTcl escrito sem usar UML

O objetivo dos *scripts* OTcl aplicando UML e sem UML é calcular o MOS de segmento de chamada. Os *scripts* OTcl foram executados no *ns-2*, versão 2.27, e apresentaram os seguintes resultados nas Tabelas 4.1 e 4.2.

Tabela 4.1: Resultado do script OTcl usando UML

```
ykashimura@linux:~/Documents/ns-allinone-2.27/ns-2.27/ns-tutorial/examples> ns Cenario_GenVoice83a.tcl
num_nodes is set 30
INITIALIZE THE LIST xListHead
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
Resultado =>  4.4122698323539522
```

Tabela 4.2: Resultado do script OTcl sem usar UML

```
ykashimura@linux:~/Documents/ns-allinone-2.27/ns-2.27/ns-tutorial/examples> ns teste_2.tcl
num_nodes is set 30
INITIALIZE THE LIST xListHead
Starting simulation...
Creating call 1 ... source 1
channel.cc:sendUp - Calc highestAntennaZ_ and distCST_
highestAntennaZ_ = 1.5,  distCST_ = 550.0
SORTING LISTS ...DONE!
Call 1 ... MOS --> 4.4122698323539522

Creating call 2 ... source 1
Call 2 ... MOS --> 4.2558652467448912

MOS medio --> 4.3340675395494213
```

4.5 Grau de complexidade na geração dos *scripts* OTcl

O cenário de uma rede *Ad Hoc* pode apresentar um grau de abstração do simples ao mais detalhado. A grau de complexidade do cenário influencia no resultado do *script* OTcl. Em cenários envolvendo elementos comuns e com comportamento de eventos simples, o *script* OTcl usando UML produz um resultado similar ao *script* OTcl sem uso da UML. Em

cenários envolvendo elementos comuns, mas com comportamentos de eventos atípicos, o *script* OTcl usando UML não tratará esses eventos. É necessário modelar os aspectos atípicos.

Capítulo 5

Considerações finais

Este trabalho apresentou um método para auxílio do ensino de simulação de sistemas de redes de computadores utilizando modelos de redes representados em diagramas UML. Através de análise e estudo em vários exemplos de *scripts* de simulação conduziu a definição dos estereótipos que caracterizam elementos e comportamento de eventos que foram aplicados em diagramas UML para modelar um cenário de simulação para tráfego VoIP em uma rede *Ad Hoc*. A metodologia para elaborar os elementos em modelos UML e definição dos estereótipos possibilitou a geração de *script* OTcl para um cenário de rede *Ad Hoc*, a partir de do padrão XMI produzido em uma ferramenta de modelagem ArgoUML.

Este trabalho envolve três abordagens: Redes de Computadores, Engenharia de Software e ensino pedagógico. O desenvolvedor do método deve buscar conhecimentos na área de Redes de Computadores, especificamente em redes *Ad Hoc*, aplicar os conceitos das técnicas de modelagem empregadas na Engenharia de Software para auxiliar o processo de extrair requisitos importantes sobre uma rede *Ad Hoc*, e se preocupar com a parte pedagógica, elaborando uma metodologia com as etapas para a construção do cenário usando uma ferramenta de modelagem em UML.

Alguns comentários serão apresentados quanto as dificuldades, limitações e sugestões para trabalhos futuros.

5.1 Dificuldades encontradas no desenvolvimento do método de ensino apoiado em UML

A topologia de uma rede *Ad Hoc* pode constar poucos nodos móveis, mas geralmente, os alunos fazem experimentos usando uma topologia de uma rede densamente povoada.

Observou-se que não é adequado para o aluno descrever a criação de cada nodo móvel em UML quando essa topologia da rede é densamente povoada. A solução foi criar um estereótipo «*Gerador_Cenário*» para que programa tradutor OTcl pesquise na biblioteca do *ns-2* um programa gerador de topologias chamado de *scen-gen*. Esta solução também foi aplicado para situação de uma rede com muitos tráfegos, criando um estereótipo «*Gerador_Tráfego*».

Aplicação dos parâmetros usados nos elementos em estereótipos diferentes. A solução foi usar o fluxo de objeto. Os valores dos parâmetros foram especificados no mecanismo chamado *tagged value* (valor etiquetado) no ArgoUML. Esse mecanismo possui dois campos: *tag* e *value*. O nome do campo e o valor do parâmetro dos elementos, definidos nas tabelas descritas na Seção 3.5, foram aplicados em *value* desse mecanismo.

Mapear todos os possíveis comportamentos de eventos para cada elemento. Em tráfego VoIP para calcular o MOS, observou-se que o aluno pode fazer experimentos em que a construção dos comportamentos dos eventos podem ser de várias formas. A solução foi usar regra básica. A metodologia descrita no Capítulo 4 apresentou regras básicas para descrever tais comportamentos de eventos para calcular o MOS em tráfego VoIP.

5.2 Limitações do método de ensino apoiado em UML

O método produz o *script* OTcl para os elementos que foram mapeados em UML. O desenvolvedor pode mapear e classificar um novo estereótipo para um elemento novo para o *ns-2*, implementado-o no programa tradutor para gerar a sintaxe no *ns-2* para o novo elemento no *script* OTcl. Porém, o novo elemento deve ser criado nas hierarquias de classes em C++ e no interpretador OTcl. A execução do *script* OTcl produzirá erro devido a ausência das classes referentes ao elemento novo nas bibliotecas do *ns-2*. Um exemplo foi comentado no Capítulo 4 descrevendo os procedimentos para criar um novo agente.

Variáveis como expressões matemáticas e inclusões de mensagens não foram tratados e mapeados, pois a idéia é capturar os aspectos mais relevantes do cenário de rede *Ad Hoc*.

5.3 Sugestões para trabalhos futuros

Devido ao recurso do XMI e o mecanismo de extensão como criação de novos estereótipos, a ferramenta ArgoUML mostrou-se adequada para desenvolver o método para auxiliar o ensino de simulação. Além disso, o ArgoUML permite adicionar novas funcionalidades para atender as necessidades específicas de um projeto. Sendo assim, algumas melhorias para futuros trabalhos podem ser feitas.

a) adicionar novos diagramas, em forma de ícones que representem os elementos de um cenário com os seus respectivos estereótipos, observando que o ArgoUML também deverá produzir esses elementos no formato XMI.

b) implementar modelos em UML para outros tipos de redes de computadores tais como, satélite, roteamento hierárquico e roteamento multicast, desenvolvidos no Network Simulator.

c) criar regras formais que descrevam o mapeamento de modelos UML para sintaxe do *ns-2* para tradutor OTcl.

Referências Bibliográficas

[1] CÂMARA, Daniel. *Estudo de Algoritmos de Roteamento para Redes Móveis Ad Hoc*. 108p.. Dissertação (Mestrado em Ciência da Computação) Departamento de Ciência da Computação, Universidade Federal de Minas Gerais. Belo Horizonte, 2000.

[2] JOHNSON, David B.; MALTZ, David A.. *Dynamic Source Routing in Ad Hoc Wireless Networks*. In: Mobile Computing, edited by Tomasz Imielinski and Hank Korth, Chapter 5, pp. 153-181, Kluwer, 1996.

[3] PEREIRA, Ivana Cardial de M. *Análise do Roteamento em Redes Móveis Ad Hoc em Cenários de Operações Militares*. 107p.. Dissertação (Mestrado em Ciência em Engenharia Elétrica) Programa de Pós-Graduação de Engenharia, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2004.

[4] The ns Manual. *The VINT Project*. Disponível em: <http://www.isi.edu/nsnam/ns/>. Acesso em : 12 set. 2004.

[5] GREIS, Marc. *Tutorial for the Network Simulator "ns"*. Disponível em : <http://www.isi.edu/nsnam/ns/tutorial/index.html>. Acesso em : 10 set. 2005.

[6] TANEMBAUM, Andrew. S. *Redes de Computadores*. 3 ed. Rio de Janeiro: Editora Campus, 1997. 923p.

[7] CLAYPOOL, Mark; CHUNG, Jae. *Ns by Example*. Worcester Polytechnic Institute, 2002. Disponível em : <http://nile.wpi.edu/NS/>. Acesso em : 09 abr. 2006

[8] ABINADER JÚNIOR, Fuad M.. *Geração de modelos de protocolos em notação SIDE/SMURPH a partir de diagramas UML*. 112p.. Monografia (Graduação em Ciência da Computação) Departamento de Ciência da Computação, Universidade Federal do Amazonas. Manaus, 2003.

[9] NAYLOR, Thomas H. et al. *Técnicas de Simulação em Computadores*. Rio de Janeiro: Editora Vozes, 1971. 398p.

[10] PETERSON, Larry L.; DAVIE, Bruce S. *Computer Network: a system approach*. 2nd ed. California : Morgan Kaufmann, 2000. 748p.

[11] WALL, Larry; C HRISTIANSEN, Tom; ORWANT, Jon. *Programming Perl*. 3th ed. California : O'Reilly, 2000. 1070p.

[12] VASQUES, Alan T. ; ESTEVES, Rafael P.; ABELEM, Antônio J.G. *Simulação de Redes de Computadores utilizando o Network Simulator*. XI Semana de Informática da UFPA- SEMINF. Pará, 2004. 54p.

[13] XMI. *MOF 2.0/XMI Mapping Specification, v2.1* Setembro, 2005. Disponível em: <http://www.omg.org/docs/formal/05-09-01.pdf>. Acesso em : set.2006

[14] FURLAN, Jose D. *Modelagem de Objetos através da UML: análise e desenho orientados a objeto*. 1 ed. São Paulo : Makron Books, 1998. 344p.

[15] Tigris. Projeto ArgoUML. Disponível em: <http://argouml.tigris.org>. Acesso em : 10 out. 2005

[16] BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. *UML - Guia do Usuário*. 1 ed. Rio de Janeiro : Campus, 2000. 504p.

[17] ALTMAN, Eitan; JIMÉNEZ, Tania. *NS Simulator for beginners – Lectures notes 2003-2004*. Venezuela, 2003. 146p.

[18] BAUMEISTER, Hubert et al. *Extending Activity Diagrams to Model Mobile Systems*. In: In M. Aksit, M. Mezini, and R. Unland, editors, Objects, Components, Architectures, Services, and Applications for a Networked World. International Conference NetObjectDays, Erfurt, Germany, 2002. v.2591, pp.278-293.

[19] BOGGS, Wendy; BOOGS, Michael. *Mastering UML com Rational Rose 2002*. 1.ed. Rio de Janeiro : Editora Alta Books, 2002. 625p.

[20] MATEUS, Geraldo R.; LOUREIRO, Antônio A. F. *Introdução à Computação Móvel*. Belo Horizonte, 2005. 189p.

[21] GUEDES, Gilleanes T.A. *UML Uma Abordagem Prática*. 1.ed. São Paulo : Editora Novatec, 2004. 320p.

[22] Gast, M.S. 802.11 Wireless Networks-The Definitive Guide. O'Reilly. April, 2002. 464p.

[23] SVG - XML Graphics for the Web. Disponível em: <http://www.w3.org/Graphics/SVG/> . Acesso em : 05 maio 2006.

[24] CHUAH, C.N. *Modelo markoviano para geração de tráfego de voz*. Arquivo voice.tcl, versão produzida em 21 out. 1998. Disponível em: <http://www.ece.ucdavis.edu/~chuah/classes/EEC173A/lab/Proj2/> . Acesso em : 01 maio 2006.

[25] NETO, A.G de A et al. ***Impacto da mobilidade na qualidade de serviço de VoIP em redes Ad Hoc***. Artigo. Amazonas, [2003?]. Arquivo pdf.

[26] CHIOSSI, T.C. ; CARVALHO, A.M.B.R. *Introdução à Engenharia de Software*. 1 ed. São Paulo : Editora da Unicamp, 2001. pp. 69-108.

[27] GERLA, M. *Ad Hoc Network. Ad Hoc Network – Technologies and Protocols*, Edited by Prasant Mohapatra and Srikanth Krishnamurthy, Springer Science Business Media Inc. Chapter 1, pp. 1-22, 2005.

[28] PARK, V.; MACKER, J. P.; CORSON, M.S. *Applicability of the Temporally-Ordered Routing Algorithm For Use in Mobile Tactical Networks*. Disponível em: <http://www.tang.itd.nrl.navy.mil/5522/pubs/pdf_papers/**tora**_milcom98.pdf>. Acesso em : 17 de set. 2006.

[29] PERKINS, C. E.; ROYER, E. M. *Ad hoc on demand distance vector (AODV) routing.* Internet Draft, August 1998.

[30] PERKINS, C., E BHAGWAT, P. *Highly dynamic destination-sequenced distance vector routing (DSDV) for mobile computers*. In ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications, August, 1994. pp. 234–244.

[31] Padrão 802.11. *Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*. IEEE Standard 802.11, 1999. Disponível em: <http://standards.ieee.org/getieee802/802.11.html>. Acesso em : Set.2006.

[32] SILVA, A. M.; MORAES, L.F.M.. *Modelo de Mobilidade para Representação de Cenários de Ambientes Fechados*. In: Simpósio Brasileiro de Redes de Computadores. Fortaleza, maio 2005.

[33] CARVALHO, Leandro S.G.. *Implementação do Modelo E para Avaliação Objetiva da Qualidade da Fala em Redes de Comunicação VoIP*. 184p. Dissertação (Mestrado em Ciência da Computação) Departamento de Ciência da Computação, Universidade Federal do Amazonas. Manaus, 2004.

[34] SESAY, Samba; YANG, Zongkai; QI, Biao; HE, Jianhua. *Simulation Comparison of Four Wireless Ad Hoc Routing Protocols*. In: Information Technology Journal. Asian Network for Scientific Information, v. 3, pp. 219-226, 2004.

[35] BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. *UML Reference Manual*. 1 ed. Addison Wesley. 1999. 512p.

[36] KRAUS, Andrea; KOCH, Nora. A Metamodel for UWE. In: Technical Report 0301, Ludwig-Maximilians-Universität München, January, 2003. 20p.

[37] NsObject Class Reference. Disponível em: <http://www-sop.inria.fr/planete/software/ns-doc/ns-current>. Acesso em: 13 out. 2006

Apêndice A

A ferramenta de modelagem ArgoUML

Este Apêndice contém informações sobre o ambiente da ferramenta de modelagem ArgoUML mostrando como usar os diagramas UML.

O ArgoUML é uma ferramenta para uso na análise e projeto de software orientado a objeto baseada na UML, que possue contribuições de pesquisadores e estudantes de várias partes do mundo. Esta ferramenta é similar a muitas outras vendidas comercialmente como ferramentas para modelagem de software. Porém, o ArgoUML tem algumas distinções das demais ferramentas tais como :

a) É uma ferramenta cujos códigos-fonte está aberto para a comunidade de pesquisadores e estudantes que podem usar a estrutura do ArgoUML para demonstrar o desenvolvimento e evolução das ferramentas de modelagem na UML. Os códigos fontes são controlados através do CVS (Concurrent Versioning System), um sistema gerenciador de versões.

b) É uma aplicação desenvolvida totalmente em linguagem Java.

c) Tem suporte para estender padrões UML, XMI, SVG e OCL como exemplos. SVG (Scalable Vectorial Graphics) [23] é uma linguagem XML (Extended Market Language) para descrever de forma vectorial desenhos e gráficos bidimensionais, de forma estática, dinâmica ou animada. OCL é uma linguagem usada para definir restrições dentro do metamodelo. As definições dos tipos para cada restrição devem ser consistentes entre si. Por exemplo, não se pode comparar um tipo inteiro com um tipo array. UML e XMI foram comentados no Capítulo 2, Seção 2.3.

O ArgoUML é uma ferramenta que está em constante desenvolvimento com várias versões disponíveis. Esta dissertação adotou a versão 0.19.8 pois possui a extensão do padrão XMI e uma propriedade que permite a inclusão de novos estereótipos, evitando a implementação deles no código fonte do ArgoUML. Estes estereótipos serão usados nos diagramas de classes e estados de ação dos cenários em redes *Ad Hoc*. A versão 0.19.8 e o manual de documentação podem ser obtidos no endereço http://argouml.tigris.org .

A Figura A.1 mostra a interface principal do ArgoUML.

Figura A.1: Intefarce principal do ArgoUML

A interface do ArgoUML possue componentes como menu de diagramas, janela de edição dos diagramas, janela de edição do projeto e janela de edição para conFiguração do artefato. O usuário escolhe um dos diagramas que são mostrados na barra de seleção dos diagramas UML. Ao escolher o diagrama que se deseja trabalhar, a janela de edição dos diagramas é aberta com os artefatos que podem ser usados para o diagrama escolhido. Um artefato é uma representação gráfica, por exemplo, classe, agregação, associação, ator, caso de caso, estado de ação ou fluxo de objeto.

Na janela de edição de configuração do artefato é onde são preenchidos os detalhes do artefato. A aba *propriedades (properties)* contém a definição do nome e as características específicas de cada artefato. A janela de edição do projeto mostra a estrutura hierárquica de todos os artefatos dos diagramas envolvidos no projeto. A barra de menu mostra itens que o usuário pode criar um novo projeto, salvar o projeto e gerar código XMI do projeto para citar os mais utilizados.

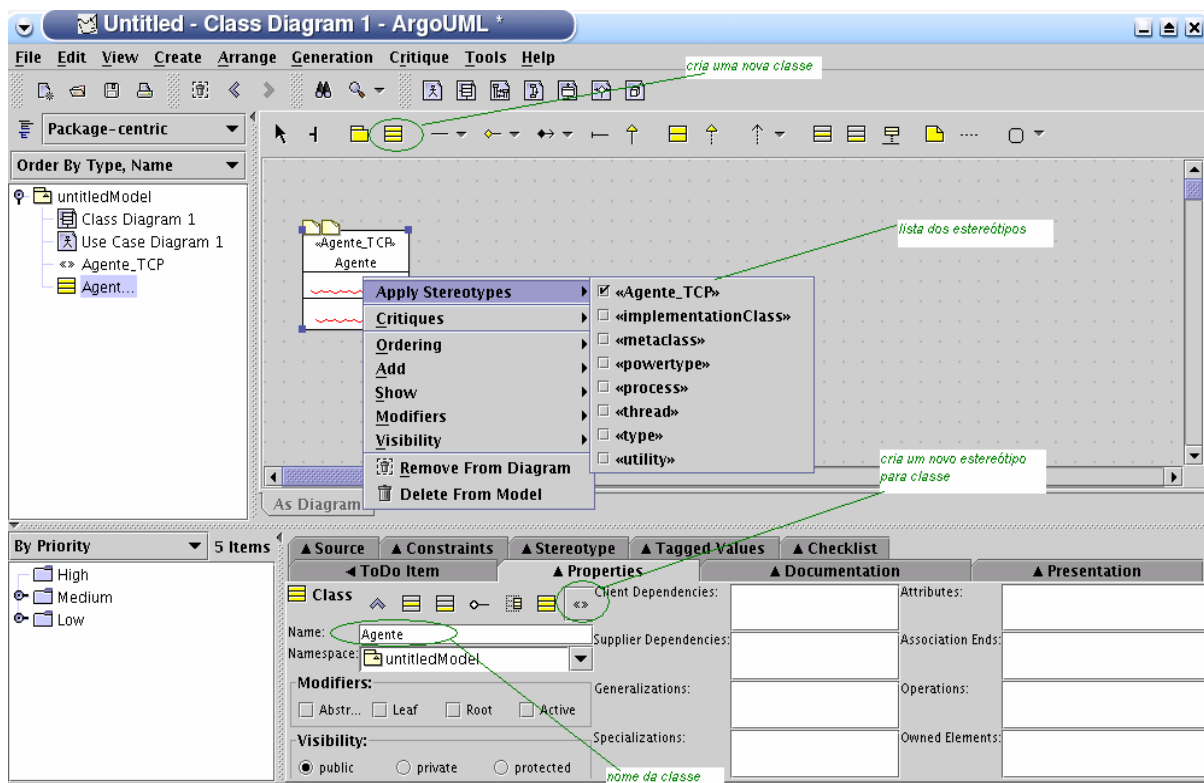



Figura A.2: Propriedades definidas para uma classe

A Figura A.2 mostra um artefato classe criado na janela de edição do diagrama de classes escolhido na barra de seleção de diagramas da UML. Na aba *propriedades* foram preenchidos o nome da classe *Agente* e um estereótipo novo «*Agente_TCP*». Um diagrama UML pode ter vários estereótipos. Um estereótipo pode ser usado em artefatos diferentes dentro do mesmo diagrama. Para visualizar a lista desses estereótipos, basta selecionar e acionar um botão de *ajuste* no artefato do diagrama que está trabalhando. O botão de ajuste é uma terminologia usado no ArgoUML e refere-se ao botão direito do mouse.

Tendo identificado as classes do cenário e representado no diagrama de classes, o usuário começa a desenhar as atividades.

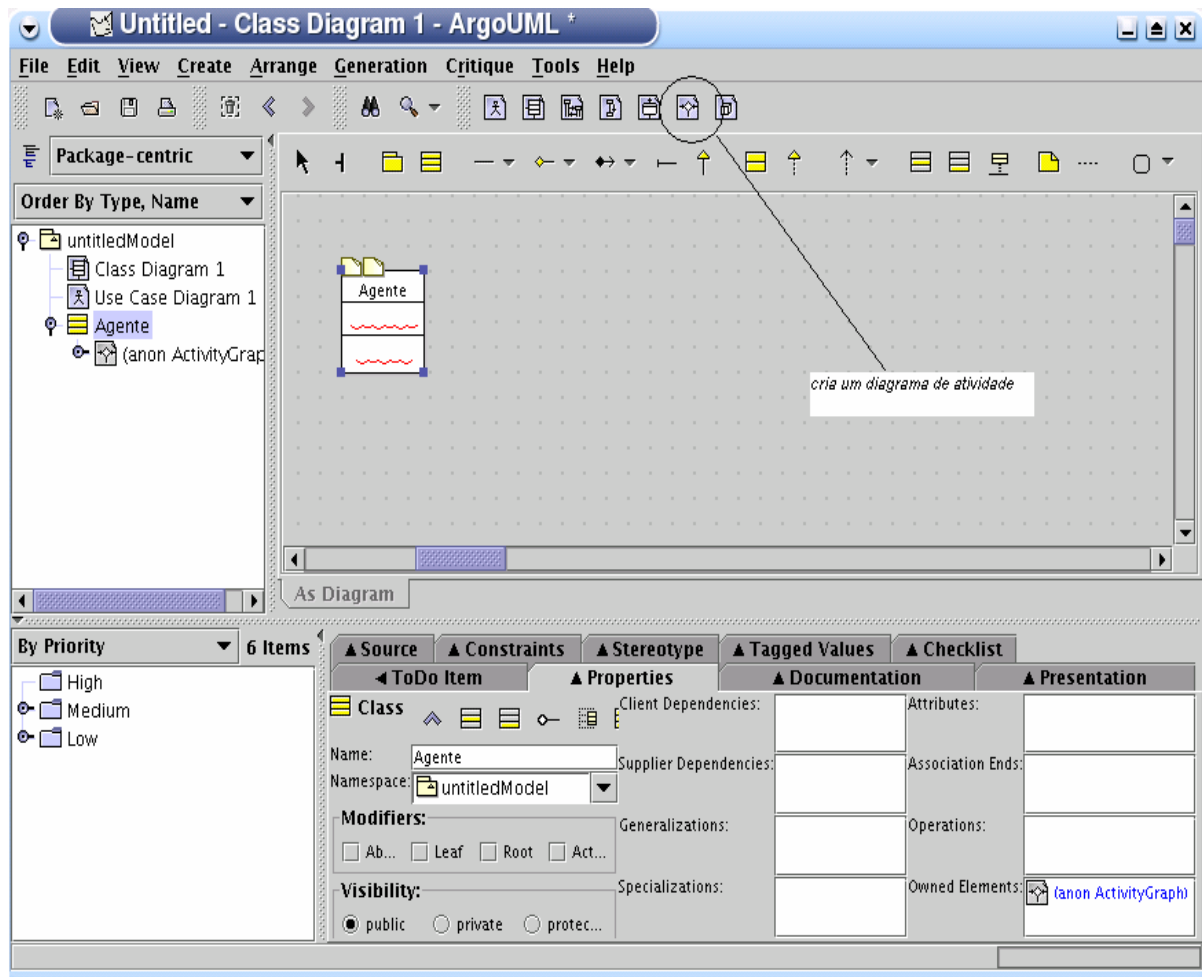



Figura A.3: Um novo diagrama de atividade

A Figura A.3 mostra o ícone para criar um novo diagrama de atividade na barra de seleção de diagramas UML, a partir de uma classe selecionada em um diagrama de classes. O usuário pode criar vários diagramas de atividades para uma mesma classe. Por exemplo, um diagrama de atividade descrevendo um tráfego usando serviço CBR conectado a um agente UDP e uma outra atividade descrevendo outro tipo serviço FTP conectado a um agente TCP.

Após criar um novo diagrama de atividade, o usuário detalha a atividade adicionando os estados de ação e os fluxos de objetos.

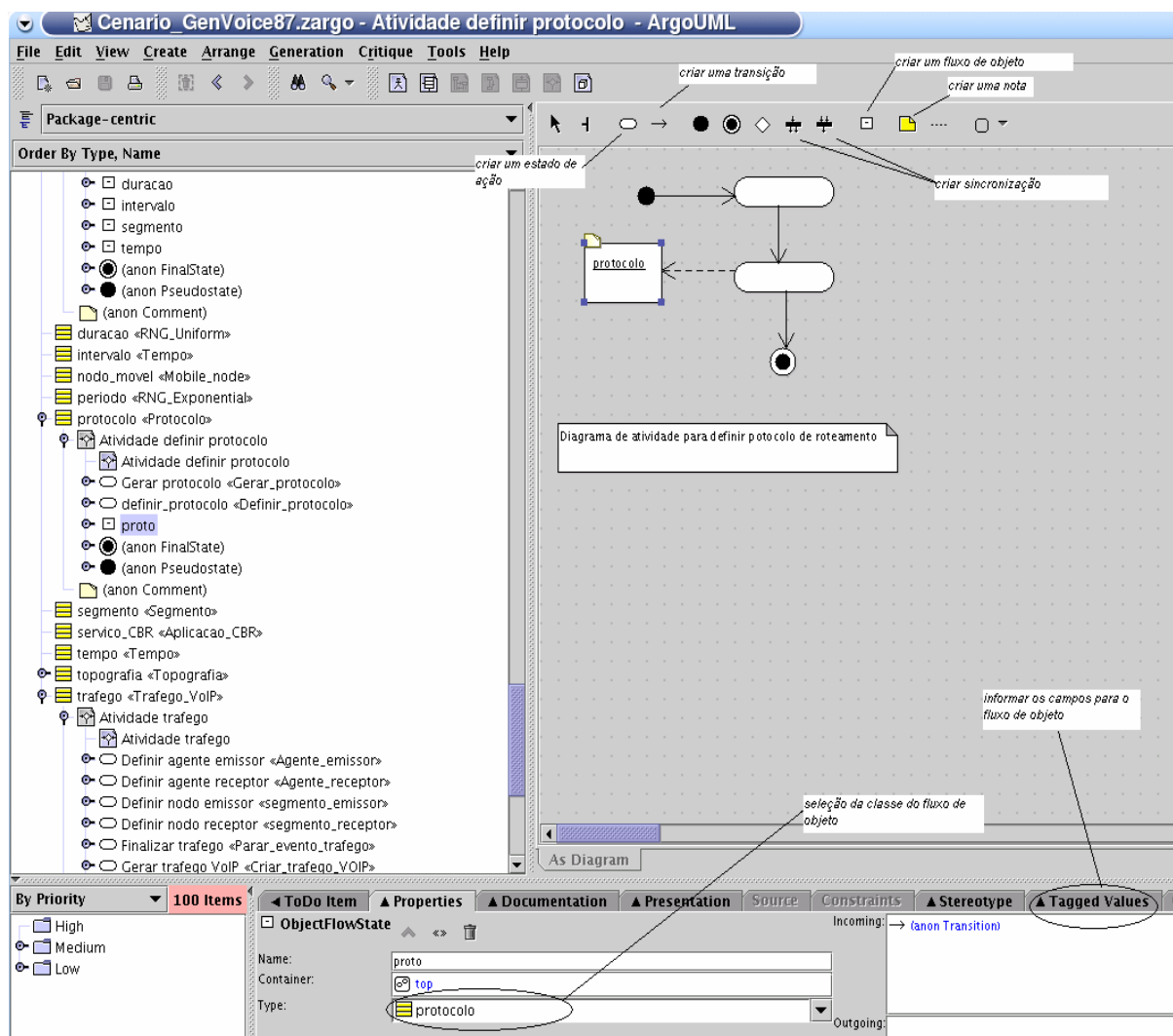
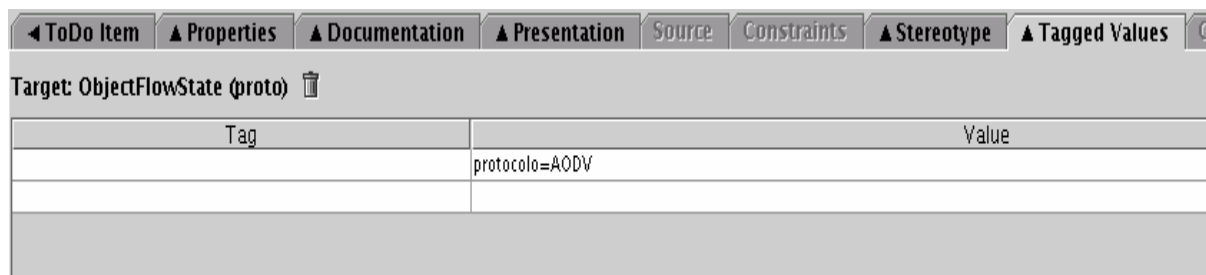



Figura A.4: Atividade para especificar protocolo de roteamento

A Figura A.4 mostra uma janela com os artefatos do estado de ação, estado de ação inicial, estado de ação final, fluxo de objeto e transição que são usados para descrever uma atividade. O usuário pode adicionar os artefatos para representar o seu cenário. Como exemplo, a Figura A.4 mostra uma atividade para representar o protocolo de roteamento usado na rede *Ad Hoc*. Ao selecionar o fluxo de objeto, o ArgoUML mostra uma aba chamada *properties* onde o usuário preenche um nome e escolhe a classe que se refere o objeto, e mostra uma aba chamada *tagged value* onde são informados os valores para objeto da classe escolhida.

ToDo Item | Properties | Documentation | Presentation | Source | Constraints | Stereotype | Tagged Values

Target: ObjectFlowState (proto)

Tag	Value
	protocolo=AODV

Figura A.5: Valor do objeto para definir o protocolo de roteamento

A Figura A.5 mostra o valor do objeto que define o protocolo de roteamento usado na rede na aba do *tagged value* no fluxo de objeto da classe protocolo.

Como em qualquer outra ferramenta, salve o projeto periodicamente selecionando *File* --> *save as*. O arquivo do modelo é salvo com a extensão *.zargo*. Para salvar o projeto no padrão XMI, o usuário pode selecionar *File* --> *Export XMI...* no ArgoUML.

Apêndice B

O código XMI gerado para cenário do estudo de caso

Este Apêndice mostra um exemplo do código XMI gerado pela ferramenta ArgoUML para o cenário do estudo de caso. Todos os artefatos como os estereótipos, estados de ação, fluxos de objetos e relacionamentos descritos na interface do ArgoUML são gerados seguindo a padronização definida pela OMG. O arquivo XMI para o estudo de caso contém 2543 linhas de códigos (45 páginas) e devido ser muito extenso foram selecionadas duas classes, nodo_móvel e topografia.

```
        <UML:Class xmi.id = '.:0000000000000821' name = 'nodo_movel' visibility = 
'public'
          isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 
'false'
          isActive = 'false'>
          <UML:ModelElement.stereotype>
            <UML:Stereotype xmi.idref = '.:0000000000000822'/>
          </UML:ModelElement.stereotype>
        </UML:Class>
        <UML:Stereotype xmi.id = '.:0000000000000822' name = 'Mobile_node' 
isSpecification = 'false'
          isRoot = 'false' isLeaf = 'false' isAbstract = 'false'>
          <UML:Stereotype.baseClass>Class</UML:Stereotype.baseClass>
        </UML:Stereotype>
        <UML:Class xmi.id = '.:0000000000000831' name = 'topografia' visibility = 
'public'
          isSpecification = 'false' isRoot = 'false' isLeaf = 'false' isAbstract = 
'false'
          isActive = 'false'>
          <UML:ModelElement.stereotype>
            <UML:Stereotype xmi.idref = '.:0000000000000832'/>
          </UML:ModelElement.stereotype>
          <UML:Namespace.ownedElement>
            <UML:ActivityGraph xmi.id = '.:000000000000082F' name = 'Atividade 
definir topografia'
              isSpecification = 'false'>
              <UML:StateMachine.context>
                <UML:Class xmi.idref = '.:0000000000000831'/>
              </UML:StateMachine.context>
              <UML:StateMachine.top>
                <UML:CompositeState xmi.id = '.:000000000000082A' name = 'top' 
isSpecification = 'false'
                  isConcurrent = 'false'>
                  <UML:CompositeState.subvertex>
```

```
                    <UML:Pseudostate xmi.id = '.:0000000000000823' isSpecification
= 'false'
                      kind = 'initial'>
                      <UML:StateVertex.outgoing>
                        <UML:Transition xmi.idref = '.:000000000000082D'/>
                      </UML:StateVertex.outgoing>
                    </UML:Pseudostate>
                    <UML:ObjectFlowState xmi.id = '.:0000000000000826' name =
'topo' isSpecification = 'false'
                      isSynch = 'false'>
                      <UML:ModelElement.taggedValue>
                        <UML:TaggedValue xmi.id = '.:0000000000000824'
isSpecification = 'false'>

<UML:TaggedValue.dataValue>x=100</UML:TaggedValue.dataValue>
                          <UML:TaggedValue.type>
                            <UML:TagDefinition xmi.idref = '.:00000000000008B6'/>
                          </UML:TaggedValue.type>
                        </UML:TaggedValue>
                        <UML:TaggedValue xmi.id = '.:0000000000000825'
isSpecification = 'false'>

<UML:TaggedValue.dataValue>y=300</UML:TaggedValue.dataValue>
                          <UML:TaggedValue.type>
                            <UML:TagDefinition xmi.idref = '.:00000000000008B6'/>
                          </UML:TaggedValue.type>
                        </UML:TaggedValue>
                      </UML:ModelElement.taggedValue>
                      <UML:StateVertex.incoming>
                        <UML:Transition xmi.idref = '.:000000000000082B'/>
                      </UML:StateVertex.incoming>
                      <UML:ObjectFlowState.type>
                        <UML:Class xmi.idref = '.:0000000000000831'/>
                      </UML:ObjectFlowState.type>
                    </UML:ObjectFlowState>
                    <UML:FinalState xmi.id = '.:0000000000000827' isSpecification =
'false'>
                      <UML:StateVertex.incoming>
                        <UML:Transition xmi.idref = '.:000000000000082E'/>
                      </UML:StateVertex.incoming>
                    </UML:FinalState>
                    <UML:ActionState xmi.id = '.:0000000000000828' name = 'Gerar
topografia'
                      isSpecification = 'false' isDynamic = 'false'>
                      <UML:ModelElement.stereotype>
                        <UML:Stereotype xmi.idref = '.:00000000000008BA'/>
                      </UML:ModelElement.stereotype>
                      <UML:StateVertex.outgoing>
                        <UML:Transition xmi.idref = '.:000000000000082C'/>
                      </UML:StateVertex.outgoing>
                      <UML:StateVertex.incoming>
                        <UML:Transition xmi.idref = '.:000000000000082D'/>
                      </UML:StateVertex.incoming>
                    </UML:ActionState>
                    <UML:ActionState xmi.id = '.:0000000000000829' name = 'Definir
topografia'
                      isSpecification = 'false' isDynamic = 'false'>
                      <UML:ModelElement.stereotype>
                        <UML:Stereotype xmi.idref = '.:00000000000008BB'/>
                      </UML:ModelElement.stereotype>
                      <UML:StateVertex.outgoing>
                        <UML:Transition xmi.idref = '.:000000000000082B'/>
```

```
                        <UML:Transition xmi.idref = '.:000000000000082E'/>
                      </UML:StateVertex.outgoing>
                      <UML:StateVertex.incoming>
                        <UML:Transition xmi.idref = '.:000000000000082C'/>
                      </UML:StateVertex.incoming>
                    </UML:ActionState>
                  </UML:CompositeState.subvertex>
                </UML:CompositeState>
              </UML:StateMachine.top>
              <UML:StateMachine.transitions>
                <UML:Transition xmi.id = '.:000000000000082B' isSpecification =
'false'>
                  <UML:Transition.source>
                    <UML:ActionState xmi.idref = '.:0000000000000829'/>
                  </UML:Transition.source>
                  <UML:Transition.target>
                    <UML:ObjectFlowState xmi.idref = '.:0000000000000826'/>
                  </UML:Transition.target>
                </UML:Transition>
                <UML:Transition xmi.id = '.:000000000000082C' isSpecification =
'false'>
                  <UML:Transition.source>
                    <UML:ActionState xmi.idref = '.:0000000000000828'/>
                  </UML:Transition.source>
                  <UML:Transition.target>
                    <UML:ActionState xmi.idref = '.:0000000000000829'/>
                  </UML:Transition.target>
                </UML:Transition>
                <UML:Transition xmi.id = '.:000000000000082D' isSpecification =
'false'>
                  <UML:Transition.source>
                    <UML:Pseudostate xmi.idref = '.:0000000000000823'/>
                  </UML:Transition.source>
                  <UML:Transition.target>
                    <UML:ActionState xmi.idref = '.:0000000000000828'/>
                  </UML:Transition.target>
                </UML:Transition>
                <UML:Transition xmi.id = '.:000000000000082E' isSpecification =
'false'>
                  <UML:Transition.source>
                    <UML:ActionState xmi.idref = '.:0000000000000829'/>
                  </UML:Transition.source>
                  <UML:Transition.target>
                    <UML:FinalState xmi.idref = '.:0000000000000827'/>
                  </UML:Transition.target>
                </UML:Transition>
              </UML:StateMachine.transitions>
            </UML:ActivityGraph>
            <UML:Comment xmi.id = '.:0000000000000830' isSpecification = 'false'
body = 'Diagrama de atividade para definir a topografia&#10;'/>
          </UML:Namespace.ownedElement>
        </UML:Class>
        <UML:Stereotype xmi.id = '.:00000000000008' name = 'Topografia'
isSpecification = 'false'
          isRoot = 'false' isLeaf = 'false' isAbstract = 'false'>
          <UML:Stereotype.baseClass>Class</UML:Stereotype.baseClass>
        </UML:Stereotype>
```

Apêndice C

O código fonte para mapeamento UML para *ns-2*

Neste Apêndice é apresentado uma parte do código fonte para mapear UML para a notação no *ns-2* através do padrão XMI. O código fonte foi implementado na linguagem Perl e deve ser executado dentro do diretório *bin* da pasta *perl* informando-o no comando de seu sistema operacional (comand prompt do Windows ou comand shell no Linux). O programa que executa o mapeamento UML para *ns-2* contém 3942 linhas de códigos (70 páginas). Uma amostra desse código é listado a seguir.

O código listado contém um procedimento que pesquisa todos os estereótipos usados nos diagramas UML e um procedimento para pesquisar os estados de ação que criam nodos. Caso encontrem estes estados de ação, o programa converte a criação de cada nodo para a notação do *ns-2*.

```
#Pesquisa todas as classes e estereotipos armazenando no vetor
#Vetor que guarda a classe (id, nome) e estereotipo referente ...
 my $cIndex_class_i = 0;
 my $cIndex_class_j = 0;
 my @vClass;
 my $sTemClass;

open(FILE_AUX, "$arquivo_xmi");
while ( $line_aux = <FILE_AUX> ) {
    chomp($line_aux);


  if ($line_aux =~ /<UML:Class xmi.id = /) {
      $_ = $line_aux;
      $cIndex_class_j = 0;  #inicializa por classe

     /<UML:Class xmi.id = '.:(.*)' name = '(.*)' visibility/;
```

```
            $cIndex_class_i++;
            $cIndex_class_j++;
            $vClass[$cIndex_class_i][$cIndex_class_j] = $1;

            $cIndex_class_j++;
            $vClass[$cIndex_class_i][$cIndex_class_j] = $2;
             $sTemClass = 's';
           }
           $string1 = "</" . "UML:Class>";
            if ($line_aux =~ /$string1/) {
              $sTemClass = 'n';
            }
            if ($sTemClass =~ /s/) {   #procura o seu estereotipo referente
             if ($line_aux =~ /<UML:Stereotype xmi.idref = /) {
                 $_ = $line_aux;
                 /<UML:Stereotype xmi.idref = '.:(.*)'/;
                $cIndex_class_j++;
                $vClass[$cIndex_class_i][$cIndex_class_j] = $1;
              }
            }
         } # while - File_AUX
         close FILE_AUX;

         #Pesquisa todos os estereotipos armazenando no vetor
         #Vetor que guarda o estereotipo (id, nome)
          my $cIndex_ester_i = 0;
          my $cIndex_ester_j = 0;
          my @vEstereotipo;
          my $sTemEster;
          my $sJa_lido;
           my $id_1;
           my $id_2;
          open(FILE_AUX, "$arquivo_xmi");
         while ( $line_aux = <FILE_AUX> ) {
             chomp($line_aux);

             $sJa_lido = 'n';
         if ($line_aux =~ /<UML:Stereotype xmi.id = '.:(.*)' name =
'(.*)'isSpecification/) {
               $_ = $line_aux;
               $cIndex_ester_j = 0;  #inicializa por classe
```

```
                /<UML:Stereotype xmi.id = '.:(.*)' name = '(.*)'isSpecification/;
                 $cIndex_ester_i++;
                 $cIndex_ester_j++;
                 $vEstereotipo[$cIndex_ester_i][$cIndex_ester_j] = $1;

                 $cIndex_ester_j++;
                 $vEstereotipo[$cIndex_ester_i][$cIndex_ester_j] = $2;
                       $sTemEster = 's';
            }

            if ($line_aux =~ /<UML:Stereotype xmi.id = '.:(.*)' name = '(.*)'
isSpecification/) {
                $_ = $line_aux;
                $cIndex_ester_j = 0;  #inicializa por classe
                /<UML:Stereotype xmi.id = '.:(.*)' name = '(.*)' isSpecification/;
                 $cIndex_ester_i++;
                 $cIndex_ester_j++;
                 $vEstereotipo[$cIndex_ester_i][$cIndex_ester_j] = $1;

                 $cIndex_ester_j++;
                 $vEstereotipo[$cIndex_ester_i][$cIndex_ester_j] = $2;
                   $sTemEster = 's';
                   $sJa_lido = 's';
            }
          if ($line_aux =~ /<UML:Stereotype xmi.id = '.:(.*)' name = '(.*)'/) {
                $_ = $line_aux;
                $cIndex_ester_j = 0;  #inicializa por classe
                /<UML:Stereotype xmi.id = '.:(.*)' name = '(.*)'/;
             $id_1 = $1;
             $id_2 = $2;

                 if ($sJa_lido =~ /n/) {
                 $cIndex_ester_i++;
                 $cIndex_ester_j++;
                 $vEstereotipo[$cIndex_ester_i][$cIndex_ester_j] = $id_1;

                 $cIndex_ester_j++;
                 $vEstereotipo[$cIndex_ester_i][$cIndex_ester_j] = $id_2;
                 $sTemEster = 's';
                }
            }
```

```
            $string1 = "</" . "UML:Stereotype>";
             if ($line_aux =~ /$string1/) {
                 $sTemEster = 'n';
                  $sJa_lido = 'n';
            }
          } # while - File_AUX
close FILE_AUX;

#Pesquisa os estados de acao com estereotipo Nodo ou Mobile_node
my @vposnod = ();  my $ci = 0; my $cj = 0; my @part;
my $ordem = 1;

if ($tem_gerador_cen =~ /n/) {     #se nao tem gerador de cenario, pesquisa os
nodos nos diagramas ...
while ( $ordem < 3 ) {
open(FILE_XMI, "$arquivo_xmi");
my $line_xmi;
my $sTemActionState = 'n';  my $contador = 0; my ($i,$string2,$y);
my @part; my $sActionDirecao = 'n';   my $sActionSimula = 's';
my $cont_obj_i = 0;
my $cont_obj_j = 0;
my $sEstereo_cria_nodo = 'n'; my $sTemActGraph = 'n';       my $sTemClass = 'n';
my $id_est = ' ';
while( $line_xmi = <FILE_XMI> ) {
     chomp($line_xmi);
     if ($line_xmi =~ /<UML:Class xmi.id =/) {
        $sTemClass = 's';
     }
     if ($sTemClass =~ /s/) {
        if ($line_xmi =~ /<UML:Stereotype xmi.idref = /) {
            $_ = $line_xmi;
            /<UML:Stereotype xmi.idref = '.:(.*)'/;
            #Pesquisa se o estereotipo eh Nodo ...
           for ($i = 0; $i <= $cIndex_ester_i; $i++) {
                push @part, $vEstereotipo[$i][2];
                if ($vEstereotipo[$i][1]  =~ /$1/) {
                     $string1 = $vEstereotipo[$i][2];
              if ($ordem == 1) {
                     if (($string1 =~ /Nodo/) || ($string1 =~ /Mobile_node/)) {
                        $sActionDirecao = 's';
                           }
```

```
             }
                }
             } # for- pesquisa se o estereotipo eh Nodo ...
      }
      }

      $string1 = "</" . "UML:Class>";
      if ($line_xmi =~ /$string1/) {
            $sTemClass = 'n';
            $sActionDirecao = 'n';
           $sTemActGraph = 'n';
      }
      if ($sActionDirecao =~ 's') {
        if ($line_xmi =~ /<UML:ActivityGraph xmi.id = /) {
           $sTemActGraph = 's';
        }

        $string1 = "</" . "UML:ActivityGraph>";
        if ($line_xmi =~ /$string1/) {
             $sTemActGraph = 'n';
        }
       if ($sTemActGraph =~ 's')  {
            if ($line_xmi =~ /<UML:ActionState xmi.id =/) {
             $sTemActionState = 's';
            }
            if ($sTemActionState =~ /s/) {
               if ($line_xmi =~ /<UML:Stereotype xmi.idref = /) {
                  $_ = $line_xmi;
                  /<UML:Stereotype xmi.idref = '.:(.*)'/;
             $id_est = $1;
                  #Pesquisa se o estereotipo eh Cria_nodo ...
                 for ($i = 0; $i <= $cIndex_ester_i; $i++) {
                      push @part, $vEstereotipo[$i][2];
                      if ($vEstereotipo[$i][1]  =~ /$id_est/) {
                         $string1 = $vEstereotipo[$i][2];
                       if (($string1 =~ /Criar_nodo/) || ($string1 =~
/criar_nodo/)) {
                         $sEstereo_cria_nodo = 's';
                         $contador = 0;  #Inicializa contador...
                         }
```

```
                        if (($string1 =~ /Criar_posicao/) || ($string1 =~
/Criar_posicao/)) {
                         $sEstereo_cria_nodo = 's';
                         $contador = 0;  #Inicializa contador...
                         }
                       }
                   } # for- pesquisa se o estereotipo eh Cria_nodo ...
                     }
             }
             $string1 = "</" . "UML:ActionState>";
             if ($line_xmi =~ /$string1/) {
                   $sTemActionState = 'n';
                 $sEstereo_cria_nodo = 'n';
                 $contador = 0;  #Inicializa contador...
             }
             if ($sEstereo_cria_nodo =~ /s/) {
             if ($line_xmi =~ /<UML:Transition xmi.idref = /) {
                 $_ = $line_xmi;
                    /<UML:Transition xmi.idref = '.:(.*)'/;
                   $contador++;
                   $y = $1;

                   if ($contador == 1) {
                    my @vet = ();
                    my $c = 0;
                       ($c, @vet) = pesquisa_nodo ($y);   #Chama subrotina pesquisa
caracteristicas do nodo
                          my $sigla_node = '$ns_ node';
                          my ($cria_nodo, $cor_nodo, $ns_nodo, $id_nodo, $palavra,
$cor1, $cor2);
                          my $sigla = '$';
                          my $aspa  = '"';
                    if ($ordem == 1) {
                          #formata padrao da criacao do NODO
                          $i = 0;
                          while ($i <= $c)   {
                            push @part, $vet[$i][2];
                            if ($vet[$i][1]  =~ /id/)  {
                                     $id_nodo = $vet[$i][2];
                               $cria_nodo = "set $id_nodo [$sigla_node]";
                               print OUTPUT "$cria_nodo\n";
```

```
                              $ns_nodo = $sigla . $id_nodo;
                              #guarda no vetor vposnod, os nodos moveis para usar
a notacao initial_node no NAM
                              $ci++;
                                 $vposnod[$ci][$cj] = $ns_nodo;
                            }
                           if ($vet[$i][1]  =~ /cor/)  {
                                    $cor_nodo = $vet[$i][2];
                              $ns_nodo = $sigla . $id_nodo;
                              $palavra = 'color';
                              $cor1 = $aspa . $cor_nodo;
                              $cor2 = $cor1 . $aspa;
                              print OUTPUT "set $ns_nodo $palavra $cor2 \n";
                            }
                           if ($vet[$i][1]  =~ /label/)  {
                                    $cor_nodo = $vet[$i][2];
                              $ns_nodo = $sigla . $id_nodo;
                              $palavra = 'label';
                              $cor1 = $aspa . $cor_nodo;
                              $cor2 = $cor1 . $aspa;
                              print OUTPUT "set $ns_nodo $palavra $cor2 \n";
                            }
                          $i++;
                         } # while
                      }  #fim- if $ordem == 1
                    if ($ordem == 2) {
                         $i = 0;
                         while ($i <= $c)   {
                           push @part, $vet[$i][2];
                           if ($vet[$i][1]  =~ /id/)  {
                                    $id_nodo = $vet[$i][2];
                              $cria_nodo = "$sigla . $id_nodo";
                            }
                           if ($vet[$i][1]  =~ /x/)  {
                                    $cor_nodo = $vet[$i][2];
                              $ns_nodo = $sigla . $id_nodo;
                              $palavra = 'color';
                              $cor1 = $aspa . $cor_nodo;
                              $cor2 = $cor1 . $aspa;
                              print OUTPUT "set $ns_nodo $palavra $cor2 \n";
                            }
```

```
                            if ($vet[$i][1]  =~ /label/)  {
                                      $cor_nodo = $vet[$i][2];
                               $ns_nodo = $sigla . $id_nodo;
                               $palavra = 'label';
                               $cor1 = $aspa . $cor_nodo;
                               $cor2 = $cor1 . $aspa;
                               print OUTPUT "set $ns_nodo $palavra $cor2 \n";
                             }
                           $i++;
                          } # while
                       }  #fim- if $ordem == 2
                 }
                } #fim
             } #fim- sEstereo_cria_nodo
        }  # fim sTemActGraph
     }
} #fim-while
close FILE_XMI;
$ordem++;
} #fim-while
print OUTPUT "\n";
}  # fim-tem_gerador_cen
```